Sviluppo di uno strumento per la simulazione degli effetti di particelle ionizzanti su dispositivi integrati

Candidato:
Alfredo Nardi

Relatori:
Prof. Paolo Bruschi
Prof. Giovanni Pennelli
Ing. Franco Bigongiari

Dipartimento di Ingegneria dell'Informazione
Università di Pisa

Indice

Capitolo 1

Introduzione

I sistemi sviluppati per applicazioni spaziali richiedono una cura speciale durante tutte le fasi della produzione, dalla progettazione all'assemblaggio, dato che sono sottoposti a condizioni estreme dove la manutenzione è difficile o impossibile. L'elettronica per lo spazio non fa eccezione. I circuiti integrati devono resistere a intense dosi di radiazioni che causano pesanti effetti ed interferiscono con il funzionamento. Le conseguenze a lungo termine sono una progressiva degradazione delle performance dovuta all'accumulo di cariche negli isolanti e di alterazioni della struttura cristallina, ma anche il passaggio di una singola particella energetica può interrompere parzialmente o completamente il funzionamento o causare gravi danni permanenti.

Per fare una valutazione accurata del rischio e sviluppare tecniche di prevenzione è fondamentale poter simulare i fenomeni che si verificano all'interno di un chip sottoposto a radiazioni. Mentre il comportamento sul lungo termine dovuto all'accumulo di radiazioni può essere stimati anche con modelli circuitali semplici, gli effeti causati dal passaggio di una particella singola dipendono da molti parametri, come tipo di radiazione, energia, angolo di incidenza e punto della superficie colpito.

Con l'incremento della densità di integrazione e l'evoluzione della tecnologia la descrizione delle interazioni possibili si è complicata. Di conseguenza un modello circuitale a parametri concentrati è inadatto a rappresentare lo spettro di fenomeni possibili.

Lo scopo di questa tesi è lo sviluppo di uno strumento per simulare il comportamento dei portatori all'interno di un circuito integrato in seguito alla ionizzazione causata dal passaggio di una particella, impiegando un modello tridimensionale implementato con una libreria Python/C++ per il metodo degli elementi finiti.

Nel primo capitolo saranno introdotte le sorgenti e le caratteristiche delle radiazioni ionizzanti, insieme alle problematiche per l'elettronica.

Infine, verrà presentato un breve excursus sulle tecniche più comuni impiegate per mitigare gli effetti negativi.

Nel secondo capitolo sarà presentata una breve panoramica degli strumenti disponibili, con particolare attenzione a FEniCS, la libreria per gli elementi finiti impiegata.

Nel terzo capitolo verrà esposto il modello impiegato, insieme ad una breve introduzione sul metodo degli elementi finiti.

Nel quarto capitolo saranno mostrate le parti più significative dell'implementazione insieme ad alcuni esempi pratici.

Capitolo 2

Radiazioni ionizzanti ed elettronica per lo spazio

In questo capitolo introduce i problemi causati dalle radiazioni. Dopo le necessarie definizioni, si passerà a mostrare effetti sull'elettronica e infine su quali sono le tecniche di protezione adottate.

2.1 Definizioni e caratteristiche

Prima di presentare le problematiche associate agli ambienti radioattivi è necessario mostrare quali sono le grandezze impiegate nella *dosimetria*, la branca della fisica che si occupa del calcolo e della misura della dose di radiazioni assorbita dalla materia.

2.1.1 Dose

La più importante grandezza dosimetrica per radiazioni energetiche è la dose assorbita, definita come rapporto fra energia assorbita e la massa del corpo:

$$D = \frac{\mathrm{d}E}{\mathrm{d}m}$$

L'unità di misura della dose assorbita nel sistema internazionale è il gray (Gy), pari a $1\frac{\mathrm{J}}{\mathrm{kg}}$. L'energia rilasciata può passare agli elettroni nel materiale e causare la ionizzazione oppure essere ceduta ai nuclei sotto forma di energia cinetica. Esistono anche altri effetti meno frequenti, come l'emissione di luce visibile nel caso della fluorescenza e della fosforescenza.

Come nota storica, dato che i primi strumenti per la misura delle quantità di radiazioni ionizzanti come il contatore Geiger[4] sfruttano la ionizzazione dell'aria, esiste una grandezza detta esposizione

che è misurata come densità di carica liberata all'interno del materiale che si misura in röntgen. Di fatto, pur misurando solo la frazione ionizzante, veniva impiegata come misura della dose totale. Pur essendo concettualmente utile in alcuni contesti, il röntgen viene impiegato raramente[5].

I danni causati dalle radiazioni dipendono dal tipo di interazioni con la materia. Di conseguenza si impiegano spesso quantità diverse come la "dose equivalente", ovvero la dose espressa in funzione del danno causato invece che rispetto alla energia effettivamente liberata (come si fa con i terremoti con la scala Mercalli). Nel caso dell'uomo si impiega il rem ("röntgen equivalent in man", unico contesto in cui il röntgen è ancora impiegato) o il sievert (Sv). Come è intuibile, si tratta di una stima media degli effetti, dato che il rischio dipende anche la velocità con cui la dose è ricevuta e il tessuto colpito[2].

Nell'ambito dell'elettronica radiation hard, la dose equivalente è universalmente espressa in rad, dove 1 rad = 0.01 Gy = 0.01 J/kg, anche se talvolta è impegato il $\frac{\text{MeV}}{\text{g}}$. Per indicare la dose ionizzante si potrebbe impiegare l'esposizione come definita in precedenza, ma la convenzione è di impiegare il rad(Si), ovvero la quantità di carica generata all'interno del silicio da un rad di energia ionizzante, a cui corrisponde una concentrazione di circa $4 \cdot 10^{13}\text{cm}^{-3}$ coppie elettrone lacuna[3].

Dato che gli effetti sulla materia e sull'elettronica sono molto differenti, la dose è sempre divisa in componente ionizzante (Total Ionizing Dose, TID) e non ionizzante (Total Non Ionizing Dose, TNID).

2.1.2 LET

Una particella che attraversa uno strato di materiale cede la sua energia lungo la traiettoria. La distribuzione lineare con la quale viene ceduta è definita dal *Linear Energy Transfer*, detto anche *Linear Stopping Power*:

$$LET = -\frac{\mathrm{d}E}{\mathrm{d}x}$$

Anche se si tratta di una forza tipicamente è espressa in $\frac{MeV}{\mu\text{m}}$, unità più impiegata nella pratica. Un altro parametro d'interesse pratico è la *Mass Stopping Power*:

$$MSP = -\frac{1}{\rho}\frac{\mathrm{d}E}{\mathrm{d}x}$$

In alcuni testi la MSP è impiegata come sinonimo della LET. Si possono identificare due cause principali della forza frenante.

Interazioni a distanza con gli elettroni: il passaggio di una carica causa lo spostamento delle altre cariche libere all'interno del cristallo e di conseguenza la particella cede la propria energia cinetica agli elettroni nelle vicinanze. In alcuni testi questo contributo è detto Elettonic Stopping Power.[16]

Urto con i nuclei: una particella che colpisce un nucleo cede parte della sua energia cinetica. Nel caso di particelle neutro come i neutroni l'interazione è dovuta principalmente allo spin magnetico, ma in prima approsssimazione è assimibilabille ad un urto classico.[17]

Il primo effetto è molto più intenso del secondo ed è nelle particelle cariche, che di conseguenza sono soggette ad una forza frenante nettamente superiore rispetto a quelle neutre.

Esiste una relazione nota fin dal 1932 con il nome di "Bethe's stopping power formula" che descrive la forza frenante a cui è sottoposta una particella carica ad alte velocità, anche se per basse energie è necessario impiegare delle correzioni aggiuntive:

$$LET = \frac{q^4}{4\pi m_e c^2 \varepsilon_0^2} \cdot \frac{nz^2}{\beta^2} \cdot \left[\ln\left(\frac{2m_e c^2 \beta^2}{I \cdot (1-\beta^2)} \right) - \beta^2 \right]$$

dove I è un parametro legato alle caratteristiche del materiale (in particolar modo al numero atomico) e β è la velocità della particella normalizzata rispetto alla velocità della luce. Gli altri valori hanno significato immediato: m_e è la massa a riposo dell'elettrone, q è la carica elementare, c è la velocità della luce, n la concentrazione di elettroni liberi nel materiale, z è il numero atomico medio e ε_0 è la permittività dielettrica nel vuoto[16]. Esistono variazioni della formula di Bethe applicabili anche a particelle neutre come i neutroni[17].

Si osserva che l'energia rilasciata da una particella carica incidente ha un massimo, detto *picco di Bragg*. La profondità del picco è determinata dall'energia iniziale della particella e dalle caratteristiche del materiale attraversato.

Anche elettroni e fotoni presentano un picco di Bragg, ma non è un fenomeno intenso come nel caso particelle cariche e pesanti come ioni e protoni.

Il picco di Bragg è il fenomeno su cui si basa l'adroterapia, una branca della radioterapia che sfrutta protoni per depositare quantità controllate di energia in profondità. L'energia ceduta causa la ionizzazione delle molecole organiche e la produzioni di radicali liberi che causano la morte delle cellule colpite. Dato che l'energia e l'intensità

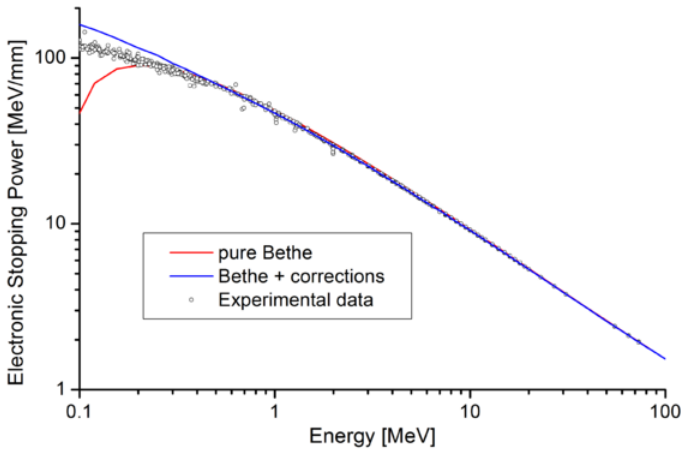


Figura 2.1: La formula di Bethe descrive con buona approssimazione il potere frenante in funzione dell'energia. Per approssimare fedelmente i dati sperimentali esistono correzioni empiriche alle energie più basse. Tratto da [88]

del fascio in ingresso può essere regolata con elevata precisione, questa tecnica è impiegata principalmente per il trattamento di tumori localizzati[6].

2.1.3 Sezione d'urto

Una grandezza legata al LET è la *sezione d'urto* (cross section), che è la relazione fra la frequenza di eventi e il flusso di radiazioni. A seconda del contesto l'evento può essere un fenomeno semplice come la ionizzazione di un elettrone oppure un fenomeno più complesso come la distruzione del componente. Il nome deriva dal fatto che "classicamente" può essere intesa come l'area di un bersaglioche genera un evento solo se colpito dalla particella-proiettile.

$$\sigma = \frac{f_{eventi}}{\Phi_{particelle}}$$

Con una LET maggiore la probabilità che si verifichi un evento di ionizzazione aumenta, anche resta un evento stocastico. In questo caso, l'andamento della cross section in funzione della LET è descrivibile come una distribuzione di Weibull[7]

$$\sigma = \sigma_{sat}\left[1 - exp\left(-\left(\frac{LET - LET_{th}}{\beta}\right)^{\gamma}\right)\right]$$

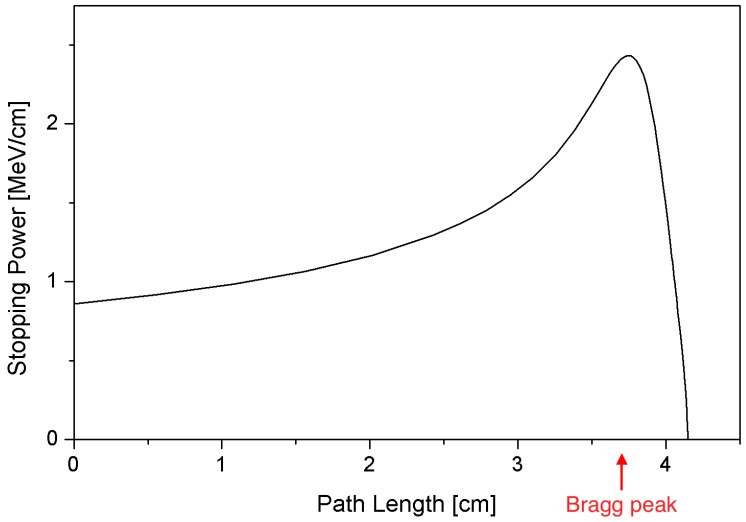


Figura 2.2: Andamento della densità lineare dell'energia dispersa da una particella α di 5.49 MeV. Adattato da [91].

dove γ e β sono parametri dipendenti dal materiale e dalla radiazione. La LET_{th} è la minima energia necessaria per generare un evento, mentre σ_{sat} è la sezione d'urto di saturazione, a cui la funzione tente asintoticamente al crescere della densità lineare dell'energia depositata.

Dato che sezione d'urto dipende dall'energia, è evidente che per una valutazione efficace del rischio in un ambiente radioattivo la dose da sola non è sufficiente, ma è necessario conoscere anche lo spettro delle energie delle radiazioni coinvolte e il loro tipo.

2.1.4 Attività

Una nota a parte merita l'*attività* (comunemente detta radioattività), ovvero il numero di eventi che si verificano all'interno di un oggetto. Tipicamente si riferisce al decadimento degli isotopi instabili all'interno di un materiale e di conseguenza è legata alla frequenza con cui sono emesse radiazioni. Si misura in Becquerel, pari ad 1 Hz.

Alcune radiazioni emesse per decadimento sono poco penetranti, per esempio le particelle α sono bloccate anche da un foglio di carta o dallo strato più esterno della pelle. Per questo motivo, in ambito pratico l'attività è espressa come flusso superficiale di particelle emesse dal materiale, dato che per la poca penetrazione le radiazioni emesse provengono solo dagli strati più esterni.

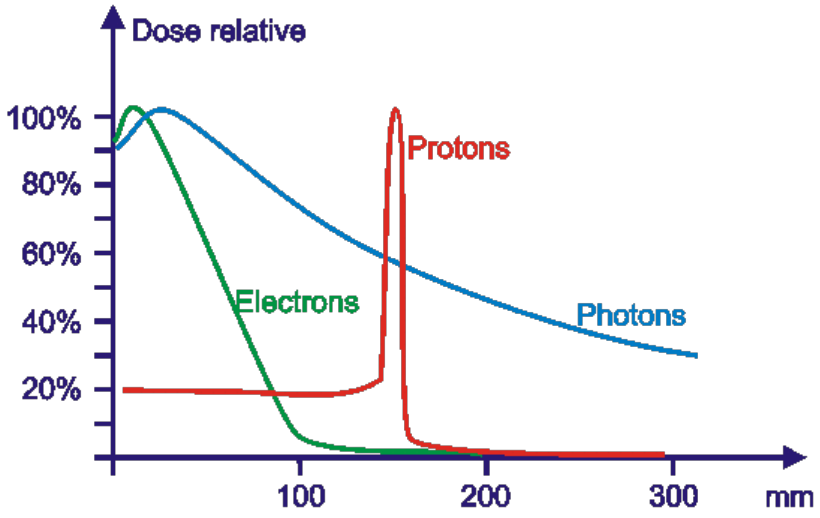


Figura 2.3: Protoni e ioni pesanti hanno un picco più ripido dei fotoni ed elettroni. Figura non in scala.

2.2 Origine

Esistono diverse sorgenti di radiazioni ionizzanti, la principale è il Sole ma altre sono esterne al sistema solare.

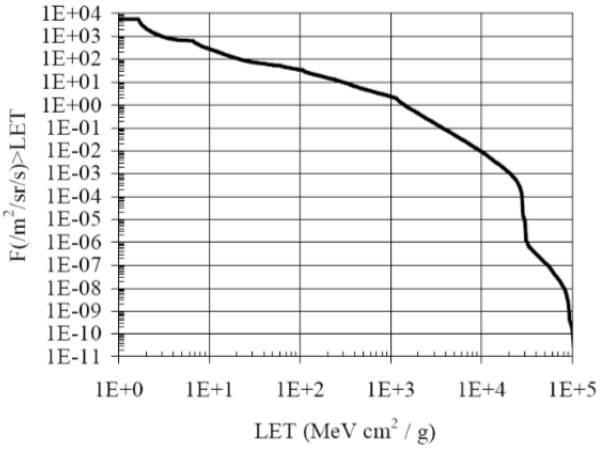


Figura 2.4: Spettro dell'energia tipico in orbita geo stazionaria. Tratto da [1]

2.2.1 Vento solare

La corona solare (lo strato più esterno del Sole) emette un flusso di particelle cariche sotto forma di plasma rarefatto, composto da protoni, elettroni e particelle α. In questa zona del Sole la temperatura è circa un milione di gradi Kelvin, nettamente superiore alla temperatura della superficie che è invece circa 5777 K. Le particelle sono a temperatura così elevata da superare la velocità di fuga del Sole e da diffondersi nel sistema solare. Il vento solare si diffonde in tutte le direzioni, anche se è meno intenso vicino a i poli a causa del campo magnetico solare. La velocità media non è costante, ma cambia al variare della distanza dal Sole. Alla distanza della Terra (1 AU) è circa 400 km/h.

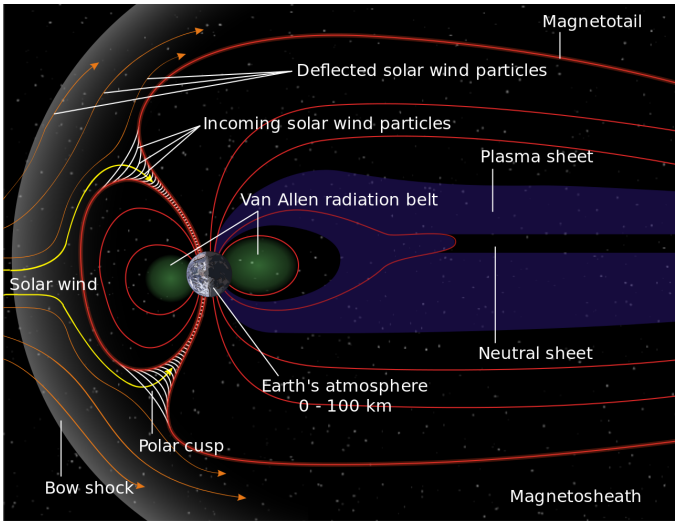


Figura 2.5: Vento solare e campo magnetico terrestre. Tratto da [90]

L'energia delle particelle emesse è compresa fra 1.5 e 10 keV. Tipicamente gli effetti a livello del mare sono fortemente attenuati dal campo magnetico terreste, ma in casi eccezionali come in presenza di tempeste solari possono essere così intensi da causare danni anche alle linee elettriche.

2.2.2 Flare

Un flare solare è una improvvisa esplosione sulla superficie, causata da variazioni rapide del campo magnetico locale che causano l'esplusione di materiale. Questi fenomeni hanno durata breve, nell'ordine dei secondi o dei minuti, ma sono abbastanza frequenti. Insieme al rilascio di ioni pesanti, si osserva spesso l'emissione di intense radiazioni distribuite su un ampio spettro di frequenze, da onde radio a raggi γ.

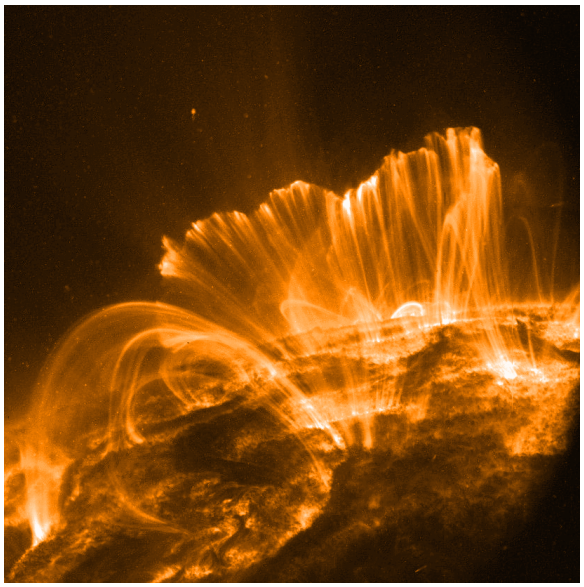


Figura 2.6: Flare solare. Tratto da [89]

2.2.3 Espulsione di massa coronale

Un altro fenomeno importante è l'emissione di materia coronale, che può durare diverse ore. Le cause sono simili a quelle dei brillamenti solari già visti, ma l'intensità del fenomeno è nettamente superiore. Le particelle cariche coinvolte possono raggiungere energie nell'ordine del MeV.

2.2.4 Raggi cosmici

Una componente minore delle particelle energetiche che colpiscono la terra proviene dall'esterno del sistema solare. Il 90% dei raggi cosmici è composto da protoni, mentre il restante è formato da particelle α e ioni pesanti. Tipicamente la loro energia è compresa fra i 10 MeV e i 10 GeV, ma sono stati osservati anche raggi cosmici con energia nell'ordine dei 10^2 EeV (10^{20} eV), detti UHECR (Ultra High Energy Cosmic Ray). Per fare un confronto, nel più grande acceleratore del mondo[1] l'energia massima raggiungibile dalle singole particelle è di soli 7 TeV, meno di un milionesimo di quella di un UHECR. Si ipotizza che alcuni raggi cosmici siano di origine extra galattica[10].

[1] Il Large Hadron Collider (LHC) di Ginevra.

2.2.5 Sorgenti terrestri

Nella crosta terrestre sono presenti isotopi che possono produrre radiazioni. Piccole quantità di torio, uranio e dei prodotti del loro decadimento sono diffusi ovunque in natura anche se raramente sono pericolosi per l'uomo, con l'eccezione del radon che è considerato la seconda causa di tumore al polmone al mondo dopo il fumo[13][14]. Le radiazioni prodotte sono poco penetranti e poco intense, con energie tipicamente inferiori ai 5 MeV, ma gli isotopi radioattivi sono particolarmente insidiosi perchè sono chimicamente indistinguibili dai corrispettivi isotopi stabili e non possono essere separati efficacemente dai metodi di purificazione normalmente impiegati nei processi microelettronici. Gli isotopi radioattivi possono accumularsi in varie fasi del processo di produzione di un componente elettronico, in particolare durante il packaging, il soldering e la metallizzazione. Per fare un esempio numerico, un wafer industriale moderno può avere una attività inferiore a 0.002 $cm^{-2}h^{-1}$. Lo stagno per saldature commerciale ha una attività di 1-5 $cm^{-2}h^{-1}$, ma nel caso di package ceramici e materiali organici può salire fino a 100 $cm^{-2}h^{-1}$[15].

Un caso più recente e insidioso dal punto di vista tecnologico è l'uso del boro, presente in quantità particolarmente alta in dispositivi sottoposti a passivazione mediante BPSG (Borophosphosilicate glass). Il boro ^{10}B reagisce in presenza di neutroni anche a basse energie liberando particelle alpha:

$$^{10}_{5}B + n \rightarrow ^{7}_{3}Li + ^{4}_{2}He$$

L'uso di BPSG non impoverito (ovvero in cui la frazione di ^{10}B non è stata ridotta mediante procedimenti chimici) causa un incremento locale di radioattività tale da non essere impiegabile anche in applicazioni commerciali non radiation hard moderne spinte come i microprocessori[8].

2.2.6 Acceleratori di particelle

Una classe a parte sono gli acceleratori di particelle, dove a seconda dell'applicazione l'elettronica deve poter resistere a intensi flussi di particelle fortemente energetiche o deve al contrario essere sensibile al passaggio di particelle che hanno poche interazioni con la materia ordinaria. Possono esservi anche intensi campi magnetici o radiazioni microonde generate per accelerare e regolare i fasci di particelle. In questo settore esiste un'intensa ricerca per aumentare la longevità dei componenti più esposti a radiazioni, ovvero i sensori. Esistono due motivi, il primo è che si tratta di oggetti progettati ad hoc con caratteristiche peculiari per i quali non esistono equivalenti commerciali (a differenza

delle componenti di controllo come, per esempio, le FPGA). Il secondo è che sostituire un componente danneggiato all'interno di un acceleratore è un'operazione di manutensione estremamente dispendiosa che può richiedere dei mesi a causa delle difficoltà nel set up (per esempio ripristinare il vuoto ultraspinto dopo l'apertura o la taratura dei fasci di particelle). Per rendere l'idea di quanto sia urgente, al termine dei lavori di incremento del flusso in corso all'LHC[2] il flusso di particelle aumenterà di 20 volte. Con un tale flusso, la vita media dei sensori attualmente impiegati si ridurrebbe a pochi mesi.

2.2.7 Particelle secondarie

Il passaggio all'interno della materia di radiazioni ionizzanti energetiche può provocare l'emissione di particelle secondarie più o meno penetranti, per nuclear spallation[3]. Nell'atmosfera i raggi cosmici interagiscono con i nuclei e producono particelle come neutroni, protoni, muoni e neutrini. Una componente è poco penetrante e viene fermata subito, mentre i neutroni possono diffondersi fino a raggiungere il terreno. Di fatto gli effetti dei raggi cosmici sono dovuti a queste particelle secondarie prodotte negli strati alti dell'atmosfera. La presenza di carbonio 14, isotopo usato per le radio datazioni dei materiali organici, è dovuto alla presenza di queste "neutron showers".

Il picco del flusso si osserva attorno ai 18 km di altezza, mentre a livello del mare si ha solo un 1/400 del massimo. Va notato che siccome i raggi cosmici sono schermati dal campo magnetico, il flusso di neutroni dipende anche dalla latitudine e dall'attività solare.

2.3 Categorie

2.3.1 Particelle cariche

Una particella carica che attraversa un cristallo può interagire con i campi elettrici presenti, fornendo energia agli elettroni che può causare l'emissione di fotoni secondari o, se l'energia è sufficiente, la formazione di una coppia elettrone lacuna. Le particelle cariche a cui di solito si fa riferimento sono i protoni, ma anche le particelle α o ioni pesanti

[2]High-Luminosity LHC upgrade (HL-LHC)

[3] Per *nuclear spallation* si intende il fenomeno per cui un nucleo libera particelle (ovvero un "frammento" del nucleo, in inglese "spall") a causa dell'urto con una particella energetica. Questo fenomeno viene usato per produrre fasci di neutroni negli acceleratori di particelle, che non avendo carica non possono essere accelerati con campi elettromagnetici.

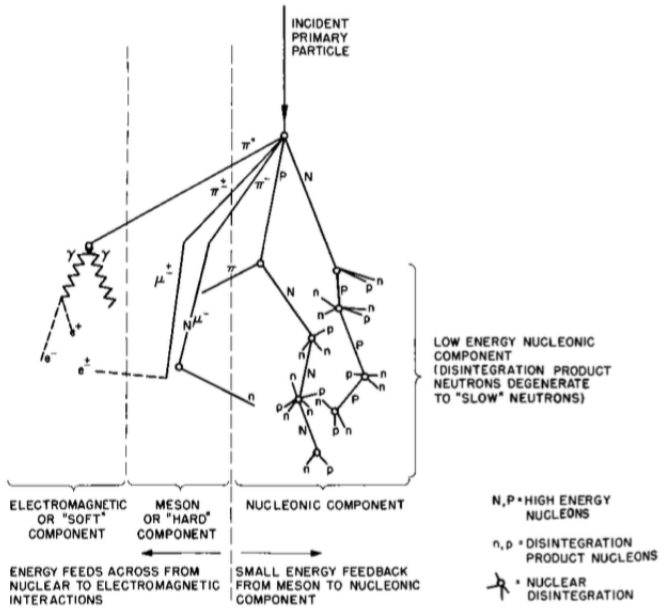
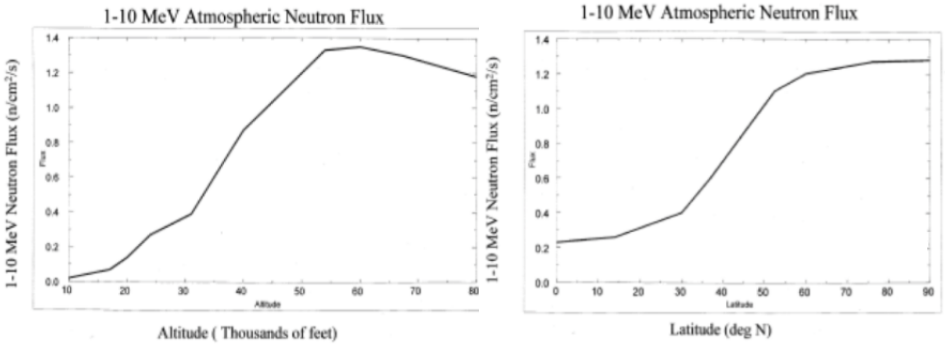


Figura 2.7: A contatto con l'atmosfera le particelle ad alta energia genera una pioggia di particelle secondarie. Tratto da [1]

hanno un comportamento simile, pur essendo meno penetranti a parità di energia. Esiste anche la possibilità che la particella carica urti un nucleo e ne causi lo spostamento dalla posizione originale. Questo ef-



(a) Flusso di neutroni medio in funzione dell'altitudine

(b) Flusso di neutroni a 10000 metri di altezza in funzione della latitudine

Figura 2.8: Dato che i neutroni sono prodotti da particelle energetiche che sono schermate daglis trati superiori dell'atmosfera e che sono più intense ai poli a causa del campo magnetico, il flusso di neutroni nell'atmosfera non è uniforme. Tratto da [1]

fetto solitamente è trascurato rispetto alla ionizzazione immediata, ma il meccanismo è del tutto simile a quello che avviene con le particelle neutre.

2.3.2 Particelle neutre

Dato che non vi sono interazioni con il campo elettrico, l'unico modo che ha una particelle senza carica di influenzare gli atomi del cristallo è scontrarsi con il nucleo. Poiché la probabilità di questo evento è piccola le particelle neutre hanno un'interazione molto ridotta con la materia. Nella pratica l'unica particella neutra che causa danni significativi è il neutrone, dato che di solito le altre hanno una massa troppo piccola o una vita media troppo ridotta. In caso di urto si possono identificare due situazioni.

Scattering Il neutrone cede una parte della sua energia al nucleo e la sua traiettoria viene deviata. L'energia in eccesso può essere ceduta interamente sottoforma di energia cinetica (scattering elastico) che può spostare il nucleo fuori dalla sua posizione di riposo o può essere convertita in un aumento di energia interna (scattering anelastico) che provoca il passaggio da uno stato fondamentale ad uno eccitato. Dopo un certo tempo il nucleo ritorna allo stato iniziale liberando una radiazione secondaria.

Assorbimento Il nucleo assorbe il neutrone, diventando un isotopo più pesante. Il nuovo isotopo può essere stabile o decadere dopo un tempo più o meno breve producendo altre radiazioni che possono causare la ionizzazione degli elettroni nelle vicinanze. Questo è il fenomeno che causa la formazione del carbonio 14 in atmosfera. L'azoto 14, colpito da un neutrone, libera un protone e un nucleo di carbonio 14.

2.3.3 Onde elettromagnetiche

Le onde elettromagnetiche non hanno carica e di conseguenza non disperdono energia velocemente quanto le particelle cariche. A differenza di quanto succede con le particelle neutre, la quantità di moto[4] è in generale troppo piccola per causare il displacement del nucleo dalla posizione di riposo. I fotoni a cui sono sottoposti i satelliti in orbita sono genericamente fra i 10 keV e i 10 MeV, abbastanza da interagire con gli elettroni. Per basse energie, è probabile che il fotone ceda ad

[4] Per i fotoni si ha che il momento trasportato è pari a $p = \frac{h}{\lambda} = \frac{E}{c}$.

un elettrone legato abbastanza energia da abbandonare il nucleo per effetto fotoelettrico: nel caso di un semiconduttore c'è la possibilità di generare una coppia elettrone lacuna. Se l'energia è superiore, il fotone può urtare un elettrone libero e cedere parte della sua quantità di moto per effetto Compton (Compton Scattering) come avverrebbe con un'urto elastico. Esiste infine un ulteriore fenomeno più esotico, che può verificarsi solo ad energie elevate: la creazione di coppia particella-antiparticella. Un elettrone (e la sua antiparticella speculare, il positrone) hanno una massa a riposo pari $9.1\dot{1}0^{-31}$ kg, da cui sia ha che l'energia necessaria per generare una coppia è $\frac{2m_0}{c^2} = 1.02$ MeV. Per generare una coppia protone-antiprotone o neutrone-antineutrone servono energie molto più grandi (rispettivamente 1876 MeV e 1879 MeV) e quindi è un fenomeno meno frequente.

2.4 Effetti sull'elettronica

Il passaggio di una particella energetica attraverso un circuito integrato può causare danni molteplici che dipendono fortemente dalla tecnologia impiegata. Gli effetti possono essere divisi in due categorie: danni che si accumulano nel tempo e fenomeni temporanei legati al passaggio di una singola particella, detti Single Event Effects (SEE).

2.4.1 Effetti cumulativi

La dose complessiva ricevuta da un dispositivo è separabile in frazione ionizzante e non ionizzante. La prima causa la generazione incontrollata di portatori all'interno del dispositivo, mentre la seconda causa una degradazione delle strutture.

Dose non ionizzante

L'esposizione prolungata a radiazioni produce cambiamenti permanenti nella struttura del silicio. L'accumulo di imperfezioni del reticolo cristallino causa la presenza di stati liberi con energie all'interno del gap del silicio, che aumentano la frequenza di eventi di generazione e ricombinazione e possono incrementare le correnti di perdita. Inoltre gli stati intermedi possono costituire una sorgente aggiuntiva di rumore telegrafico (o pop corn) dovuto al frequente trapping e detrapping degli elettroni. I difetti del cristallo interagiscono negativamente anche con i droganti, che possono essere compensati o anche migrare all'interno del silicio.

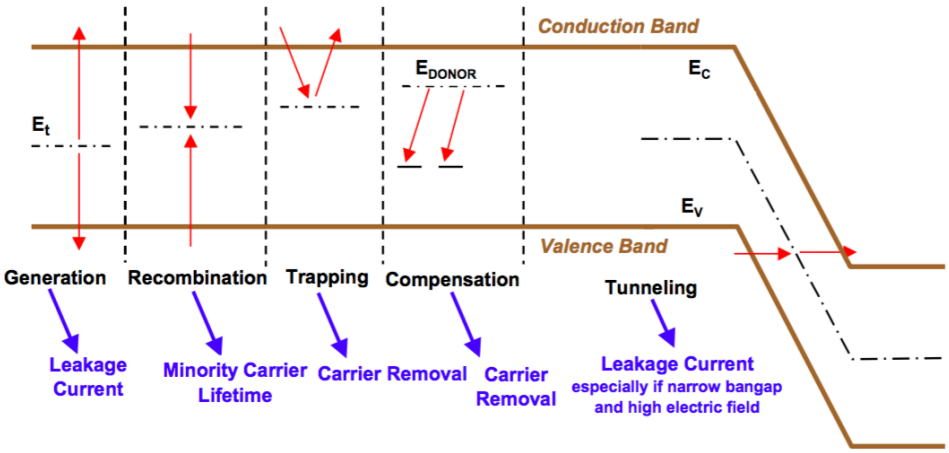


Figura 2.9: Effetti della presenza di difetti all'interno del cristallo.

A causa delle alterazioni indotte dal passaggio di particelle anche gli isolanti si degradano e perdono le loro qualità elettriche. Complessivamente l'effetto dovuto alle radiazioni non ionizzanti è assimilabile ad un invecchiamento precoce del materiale.

Se l'energia rilasciata da una singola particella è piccola, l'urto con un atomo del reticolo non causa uno spostamento di quest'ultimo ma solo un'oscillazione attorno alla posizione di riposo. In prima approssimazione, si osserva che affinchè si verifichi uno spostamento permanente dell'atomo (ovvero un danno da spostamento o Displacement Damage) è necessario che la particella incidente possieda una energia che superi una soglia minima (detta Threshold Displacement Energy).

Se è noto lo spettro dell'energia della radiazione e la dose complessiva ricevuta si può ottenere una valutazione della degradazione complessiva a cui è sottoposto il dispositivo.

Dose ionizzante

La componente ionizzante causa danni più significativi di quella non ionizzante. La generazione e l'accumulo di cariche all'interno degli isolanti causa lo spostamento delle bande energetiche. In tecnologie MOS, l'accumulo di cariche all'interfaccia causa l'alterazione delle tensioni di soglia che a loro volta possono determinare il passaggio di forti correnti di perdita, con il rischio di causare la degradazione precoce del dispositivo.

Le informazioni sono codificate nelle memorie DRAM e EEPROM sottoforma di carica accumulata. Di conseguenza, le alterazioni indotte

Orbit name	GEO	GPS	LEO	DMS
Apogee (km)	35,796	20,189	1,600	946
Perigee (km)	35,795	20,172	1,600	824
Inclination (degrees)	0	55	60	99
Dose rad(Si)/yr	6,600	59,000	17,300	1,260

Figura 2.10: Caratteristiche delle orbite standard e dose annuale di radiazioni ionizzanti. Il campo magnetico terrestre forma una protezione contro le particelle cariche di origine extraterrestre, di conseguenza a quote maggiori anche l'irraggiamento aumenta. Tratto da [1]

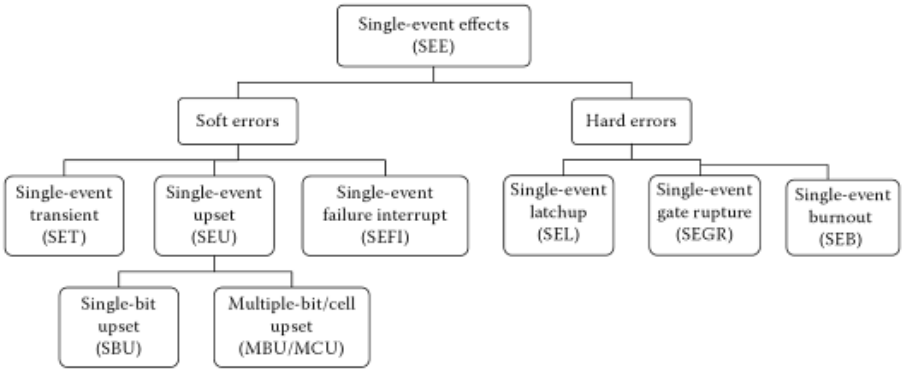
dall'irraggiamento possono corrompere i dati salvati. Nel caso delle DRAM l'effetto si può assimilare ad una riduzione del retention time e può essere parzialmente attenuato aumentando la frequenza di refresh a scapito dell'efficienza. La sensibilità delle EEPROM è particolarmente grave, dato che a differenza di altre memorie il loro contenuto deve essere mantenuto a lungo termine.

Oltre al dato in se anche la lettura è alterata (in maniera permanente o temporanea) a causa della variazione di tensione di soglia. Conoscere la dose ionizzante, ovvero la Total Ionizing Dose (TID), che un componente riceve in un certo periodo di tempo permette di valutare sia la vita media del dispositivo sia la frequenza di errori, detta Soft Error Rate (SER).

2.4.2 Eventi singoli

Anche la carica generata dal passaggio di una singola particella può indurre un malfunzionamento che a seconda della tecnologia e della natura del circuito può esaurirsi in breve tempo senza conseguenze oppure può provocare la distruzione di un dispositivo elettronico. Alcune vulnerabilità legate alla tecnologia possono essere risolte a livello architetturale, a scapito delle prestazioni.

In base agli effetti si distingue fra errori che corrompono i dati ma che non lasciano tracce permanenti sul dispositivo (detti soft error) e danni fisici ai componenti del circuito (detti hard errors).



Single Event Transient (SET)

Il passaggio di una singola particella ionizzante induce una scia di portatori all'interno del silicio che si riversa nei nodi del circuito più vicini, generando un impulso di corrente. Gli impulsi sono tipicamente schematizzati come sovrapposizione di due esponenziali (detto in letteratura Double Exponential Impulse):

$$\Delta I = \Delta I_M \left(e^{\frac{t-\Delta t_1}{\tau_1}} - e^{\frac{t-\Delta t_2}{\tau_2}} \right)$$

dove ΔI_M, Δt_1, τ_1, Δt_2 e τ_2 sono parametri dipendenti dalla energia rilasciata (o più precisamente dalla LET), dalla traiettoria, dalle caratteristiche del materiale attraversato e dalla topologia del circuito.

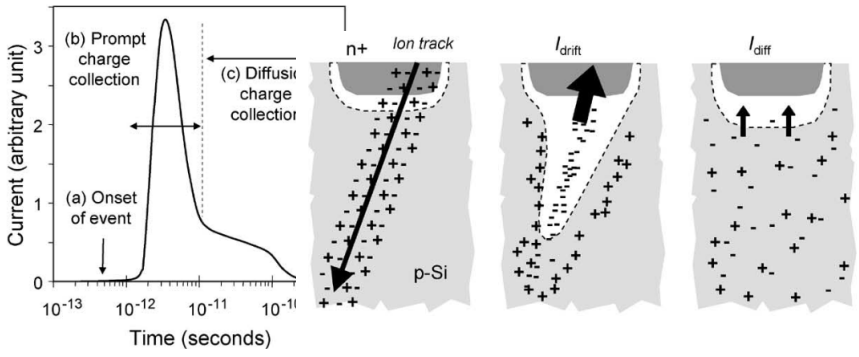


Figura 2.11: L'andamento della corrente raccolta ai nodi è approssimabile come un doppio esponenziale, con tempi che dipendono dalla fisica del dispositivo. Tratto da [26]

A seconda della traiettoria della particella, il SET può colpire più nodi vicini fisicamente provocando una serie di impulsi di corrente correlati su varie linee del circuito con ampiezze e larghezze variabili, a causa del *charge sharing*. Mentre alcune topologie sviluppate espressamente per il radiation hardening sono immuni al SET su un singolo nodo, impulsi correlati multipli possono comunque causare problemi di funzionamento.

In circuiti digitali il SET può propagarsi attraverso una rete combinatoria corrompendo il dato in uscita. Se colpisce una linea di clock può provocare un anticipo del fronte di salita portando uno o più elementi di memoria in uno stato errato.

Per il loro principio di funzionamento, gli apparati optoelettronici sono particolarmente sensibili agli effetti delle radiazioni. Notare che i sensori impiegati per rilevare il passaggio di particelle ionizzanti (nei quali il SET non è un disturbo ma è il segnale utile) hanno una struttura affine ai fotodiodi.

Single Event Upset (SEU)

Il SEU è un errore che si verifica tipicamente quando la corrente indotta dalla scia della particella è raccolta da nodi critici di elementi di memoria, con il risultato di invertire un bit memorizzato (Flip Bit).

A seconda dell'elemento colpito, gli effetti possono essere lievi (come nel caso di dati già affetti da disturbo) o avere ripercussioni a cascata gravi, come può avvenire nel caso sia colpito un registro critico.

Il SEU può essere il prodotto di un SET catturato nelle vicinanze di un fronte di clock.

In letteratura, il caso in cui solo un bit sia alterato è chiamato Single Bit Upset (SBU), per distinguerlo dalla situazione in cui un SEE corrompa più bit contemporaneamente.

Multiple Bit Upset (MBU)

Il passaggio di una particella può causare errori in più elementi di memoria fisicamente vicini, come i flip flop che contengono una parola. Un flipping multiplo sfugge ai classici algoritmi di verifica implementati nell'elettronica commerciale come il bit di parità. Anche codici di correzione dell'errore comunemente impiegati come l'Hamming code falliscono di fronte ad errore multiplo.

Multiple Cell Upset (MCU)

Elementi di memoria fisicamente vicini possono appartenere a parole diverse, quindi un singolo evento può causare la corruzione due o più registri differenti. Tuttavia, dato che il MCU causa il flipping di un solo bit nelle parole coinvolte, la correzione è più semplice rispetto al MBU.

Le EEPROM (anche Flash) possono essere alterate da un SEU, ma la probabilità di flipping multipli causati da un singolo evento (MCU e MBU) è bassa a causa dell'isolamento che esiste fra un elemento e l'altro[27].

MCU e MBU sono una conseguenza del fenomeno di charge sharing.

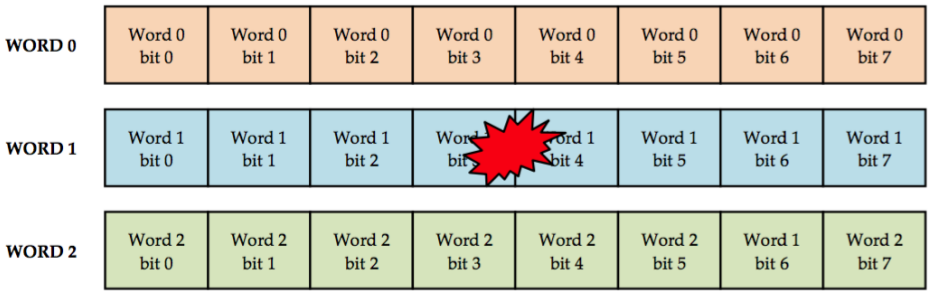


Figura 2.12: SEU su più celle adiacenti della stessa parola (MBU). Tratto da [1]

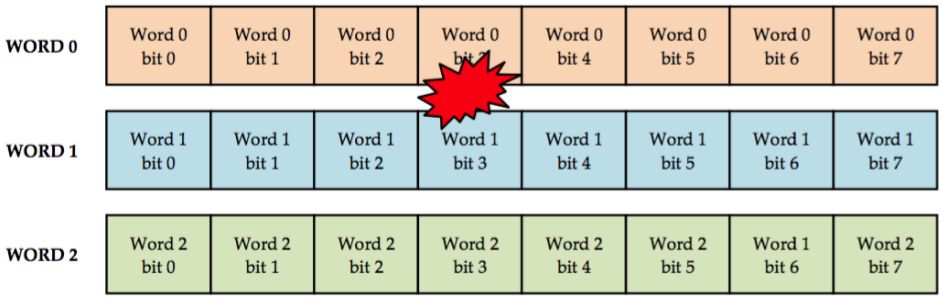


Figura 2.13: SEU su più celle di parole differenti (MCU). Tratto da [1]

Single Event Functional Interrupt (SEFI)

In macchine sequenziali complesse (come FPGA, CPU e microcontrollori) alterazioni della memoria indotte da uno o più SEU possono portare in stati non definiti, in stati di blocco o in modalità di test che interrompono il normale funzionamento. Nella categoria dei SEFI rientrano

anche alterazioni dei percorsi logici in una FPGA (che per esempio potrebbero interrompere un timer critico) o modificare il software caricato in memoria (causando per esempio un loop infinito).

Tipicamente, pur essendo potenzialmente gravi in applicazioni critiche, i SEFI sono temporanei e possono essere risolti con il reset del sistema.

Single Event Latchup (SEL)

All'interno di un circuito CMOS standard, composto da un PMOS e un NMOS adiacente, si può identificare una coppia di transistor BJT connessi come in figura. La carica proveniente SEE può essere raccolta da una base e causare l'attivazione di uno dei BJT, con la conseguente creazione di un anello in retroazione positiva.

Il forte passaggio di corrente può causare un forte riscaldamento locale e la distruzione del dispositivo. Il comportamento è identico a quello di tiristore. Notare che il latch up è un problema critico anche fuori dal costesto radioattivo, e di conseguenza negli ultimi decenni è stato affrontato anche nelle tecnologie commerciali.

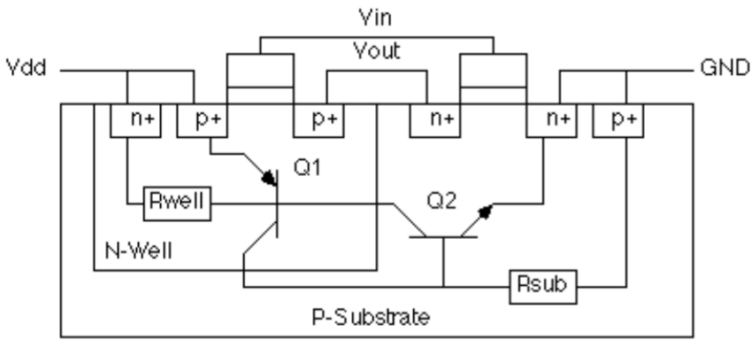


Figura 2.14: Struttura parassata che si attiva in presenza di latchup. Tratto da [1]

Single Event Burnout (SEB)

In un MOSFET verticale di potenza, può essere presente un BJT parassita costituito la giunzione nel source e dalla giunzione fra drain e source. Come visto nel Latchup, una particella ionizzante che attraversa il dispositivo può generare un piccolo impulso di corrente sulla base del transistor parassita. Se il MOSFET è sottoposto a forti tensioni, la giunzione base collettore del BJT è fortemente polarizzata in inversa e

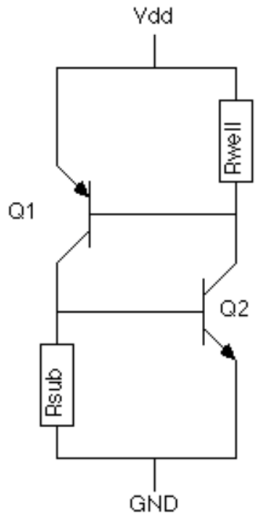


Figura 2.15: Circuito equivalente. Tratto da [1]

quindi può trovarsi in second breakdown. In queste condizioni la corrente si concentra nei punti a minore resistenza della giunzione. Dato che all'aumento della temperatura corrisponde un aumento di conducibilità, si osserva una retroazione positiva (fuga termica) che se non limitata dall'esterno può distruggere il dispositivo.

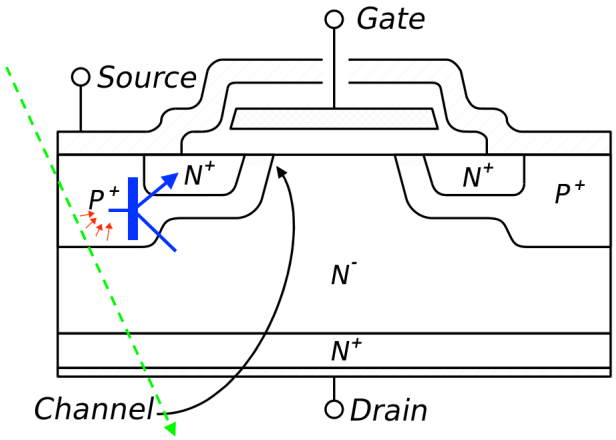


Figura 2.16: Single Event Burnout

Single Event Gate Rupture (SEGR)

Una particella ionizzante può causare la formazione di una scia di portatori all'interno dell'ossido di gate. In presenza di una forte differenza

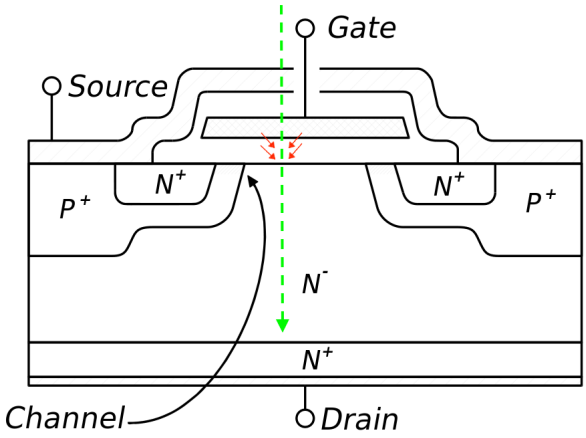


Figura 2.17: Single Event Gate Rupture.

di potenziale fra gate e canale, l'impulso di corrente libera abbastanza energia da forare l'isolante. Il SEGR è detto anche Single Event Gate Damage (SEGD).

Sia il SEB sia il SEGR sono malfunzionamenti tipici dei MOS di potenza impiegati in applicazioni aereo spaziali e sono rari in altri contesti.

2.5 Tecniche di prevenzione

Per mitigare gli effetti delle radiazioni nel corso degli anni sono state sviluppate molteplici tecniche. Tipicamente l'impiego di tecniche di radiation-hardening ha un costo in termini di overhead di spazio o di tempo ma può avere benefici secondari come la riduzione del rischio di latchup e dei consumi.

Alcuni degli effetti sono causati o aggravati dalla diffusione della carica a vari nodi del circuito (*charge sharing*) che può rendere costoso evitare alcuni fenomeni dannosi.

Una prima tecnica di difesa è schermare il dispositivo elettronico dalle radiazioni. Se applicato ingenuamente questo approccio è deleterio: la presenza di uno strato di materiale può causare l'emissione di particelle energetiche più pericolose oppure spostare il picco di Bragg di una radiazione proprio nella zona più sensibile del circuito integrato. Può anche essere difficile da mettere in pratica: molte applicazioni in ambienti radioattivi sono di tipo avionico o spaziale, con pesanti vincoli

Technology	Family	Function	SEL	SEGR	SEB	SEU	MCU/MBU	SEFI	SET
Power MOS				X	X				
CMOS, BiCMOS and SOI *except SOI	Digital	SRAM	X*			X	X		
		DRAM	X*			X	X	X	
		FPGA	X*			X	X	X	X
		Flash EEPROM	X*			X		X	
		µP / µcontroller	X*			X	X	X	X
	Mixed signal	ADC	X*			X		X	X
		DAC	X*			X		X	X
	Linear		X*						X
Bipolar	Digital					X			X
	Linear					X			X

Figura 2.18: Effetti tipici ordinati per famiglia di dispositivi. Tratto da [1]

di peso. Nel caso di impieghi a terra, l'ingombro può essere un fattore limitante, soprattutto in ambito sensoristico.

Di conseguenza, anche se a prima vista potrebbe non sembrare, lo studio di una schermatura richiede una conoscenza approfondita dell'ambiente di lavoro e una simulazione accurata delle interazioni fra il materiale e le radiazioni presenti.

2.5.1 Criteri di merito

Prima di presentare le tecniche per la prevenzione è necessario introdurre le metriche più rilevanti ai fini della progettazione. La stima degli effetti sia della TID sia della TNID è limitata alla valutazione della variazione dei parametri del materiale (come resistività, carica intrappolata e relativo threshold shift, riduzione dell'isolamento) in funzione della dose ricevuta. Pur esistendo descrizioni analitiche il loro uso non è conveniente per la complessità delle interazioni: gli effetti dell'irraggiamento dipendono anche dalla dose già ricevuta e possono addirittura curare i danni causati da esposizioni precedenti (radiation anealing).

Come è intuibile esiste una stretta dipendenza dalla tecnologia e di conseguenza l'alterazione dei parametri è ricavata con test empiricamente su esemplari di prova irraggiati in ambienti controllati. A partire da questi dati è possibile effettuare una simulazione monte carlo per

stimare la dose entro la quale il dispositivo garantisce le prestazioni (o l'affidabilità) richieste dall'applicazione[19][20].

Una eccezione è l'energia di ionizzazione del materiale: maggiore è l'energia richiesta per generare una coppia di portatori e minore è il danno dovuto a radiazioni ionizzanti. Un esempio importante è il diamante[21], che presenta una energia di ionizzazione quasi quattro volte superiore a quella del silicio e di conseguenza è impegato per sensori in esperimenti per la fisica delle alte energie come ATLAS[22]. Esistono anche studi sull'applicazione del carburo di silicio (SiC) e nitride di gallio (GaN) che presentano caratteristiche simili.

Tuttavia questo criterio è impiegabile solo in sensori per applicazioni che richiedono un'elevatissima resistenza alle radiazioni e solo se è possibile sostenere l'elevato costo di progettazione, produzione e testing di dispositivi basati su questi materiali che al momento non hanno utilizzi nell'elettronica commerciale. Esistono studi che esplorano la possibilità di impiegare grafene in dispositivi radiation hard nel prossimo futuro[9].

Esiste un ampio spettro di tecniche impiegabili per ridurre i danni da SEE su dispositivi al silicio ed esiste una ampia letteratura[23][24][25] riguardante la valutazione di queste tecniche. Storicamente, un criterio impiegato per valutare la radiation hardness di una particolare architettura o tecnologia è la carica critica Q_{crit}, ovvero la quantità minima di carica che deve essere raccolta ai nodi del circuito per causare un errore. Per esempio, nel caso di una DRAM la carica critica necessaria alla generazione di un SEU è quella necessaria a scaricare la capacità carica associata ad un elemento di memoria. Questo concetto può essere esteso anche ad altri fenomeni come SEL e SET.

Una quantità strettamente correlata è la profondità media della zona sensibile ai portatori generati, la Charge Collection Depth (CCD), che tiene conto della distribuzione della carica. A parità di carica critica, un dispositivo con profondità di raccolta minore è più resistente ai SEE rispetto a uno con profondità maggiore[28][29].

Il limite di questo approccio euristico è che nasconde gli effetti dovuti alla distribuzione della carica nel volume. Citando uno studio[26] dedicato alla simulazione degli effetti dei SEE sulle SRAM "è possibile catturare una carica totale da un evento che non causa SEU maggiore di quella ricevuta da un evento che causa un SEU". Gli effetti di un SET di conseguenza non possono essere valutati impiegando solo una simulazioni circutale.

La frequenza reale degli errori dipende dalla sezione d'urto, che è il vero indicatore della resistenza alle radiazioni. Il calcolo della sezione d'urto è molto complesso: dipende dall'evento considerato, dalle carat-

teristiche della radiazione incidente (come il tipo di particella, la sua energia e la geometria del sistema).

In applicazioni digitali la cross section può essere espressa anche come rapporto fra il flusso di radiazioni ricevuto e la frequenza di errore sul singolo bit o sulla parola, ovvero la Bit Error Rate (BER) e la Symbol Error Rate (SER).

2.5.2 Tecnologia

Le tecniche impiegate per mitigare gli effetti delle radiazioni a livello di processo tecnologico, indicate a volte con il nome di Radiation Hardening By Process (RHBP), consistono nel modificare un processo standard aggiungendo fasi di di trattameno specifiche per ridurre l'intrappolamento di cariche e quindi abbatttere i danni cumulativi legati alla TID o attenuare gli effetti degli SEE interrompendo i percorsi parassiti che causano il latchup o limitando la generazione di portatori nelle zone sensibili.

Tecnologie resistenti a TID

L'effetto più deleterio della dose ionizzante è l'accumulo di cariche nell'ossido che causa l'alterazione delle bande e il conseguente shifting della tensioni di soglia nelle strutture MOS.

In generale, l'impiego di isolanti ad alto coefficiente dielettrico (high k) ha il vantaggio di permettere una riduzione delle dimensioni del dispositivo e quindi limitare la dose totale ricevuta dal singolo elemento. Tuttavia questi isolanti sono peggiori per quanto concerne la quantità di carica intrappolata in seguito a fenomeni ionizzanti rispetto all'ossido di silicio e pertanto il loro impiego è da evitare nell'elettronica radiation hard[9].

Prendendo in esame solo l'ossido di silicio, sperimentalmente si osserva che l'esposizione a radiazioni ionizzanti causa l'accumulo di carica netta positiva.

Per ridurre l'intrappolamento delle lacune l'approccio più diretto è migliorare la qualità dell'interfaccia Si/SiO_2, che può portare altri vantaggi non legati alla radiation hardness del dispositivo, come la riduzione delle correnti di perdita o del rumore dovuto al trapping.

Senza la pretesa di presentare un elenco esaustivo, possiamo notare che alcune tecniche note in letteratura per la loro efficacia nella l'aumentare la resistenza alla TID sono nate per supplire ad esigenze non legate alla radiation hardness, come l'anealing in atmosfera inerte dell'ossido e l'ossidazione in ambiente secco controllato[31] (dry grown

thermal oxide). Anche l'impianto di azoto nel silicio per formare composti SiO_xN_y (silicon oxynitride) permette di aumentare la radiation hardness[30].

Un'altra via è quella di favorire l'intrappolamento di elettroni, in modo da riequilibrare la presenza di lacune nell'isolante e neutralizzare la carica totale. In letteratura esistono studi che dimostrano che l'impianto nel silicio di arsenico, fosforo e altri elementi prima dell'ossidazione[33] può incrementare la sezione d'urto delle trappole degli elettroni. Una tecnica particolarmente promettente basata sullo stesso principio è la diffusione di fluoro nell'ossido, seguita da anealing[32].

Oltre al costo maggiore dovuto al fatto di non essere una tecnica commerciale, un limite importante dell'impiego di impurezze nell'ossido è che l'uso di additivi può inquinare il silicio o degradare altre qualità dell'isolante.

SOI

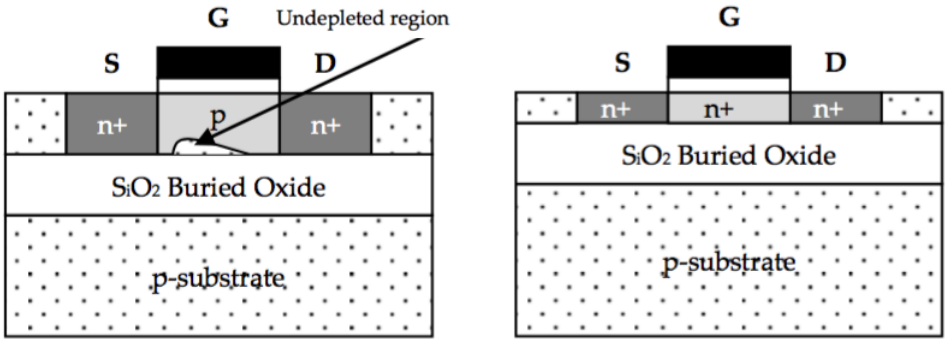
Le tecnologie basate su SOI (Silicon On Insulator)[5] sono intrinsecamente immuni al Latchup (e di conseguenza al SEL). Anziché impiegare un bulk di silicio semiconduttore come supporto meccanico ed elettrico, i transistori MOS sono costruiti sopra ad uno strato isolante. Con questa tecnica, ogni transistore possiede una well separata elettricamente dalle altre e non si può formare la struttura parassita PNPN responsabile del latchup.

La tecnologia SOI ha diversi vantaggi a livello di performance:

- l'assenza di un collegamento diretto fra i transistori limita le correnti di perdita, riducendo la potenza statica consumata;
- l'isolamento della well consente di creare sia PMOS sia NMOS isolati dal substrato indipendentemente dal drogaggio di quest'ultimo;
- i MOSFET sono immuni all'effetto body;
- lo strato isolante abbatte le capacità parassite e consente l'aumento della velocità a parità di dimensione.

Ai fini del radiation hardening, la presenza dell'isolante ha effetti contrastanti:

[5] Storicamente, la prima tecnologia SOI matura si basava sull'impiego dello zaffiro (una forma cristallina dell'ossido di alluminio Al_2O_3) come isolante. In questo caso si impiega l'acronimo SOS (Silicon On Sapphire).



(a) Partially depleted SOI.

(b) Fully depletd SOI.

Figura 2.19: Se il canale si estende fino allo strato isolante, si ha un mosfet Fully Depleted (SOI-FD). In caso contrario si ha un mosfet Partially Depleted (SOI-PD). Tratto da [1]

- si formano meno portatori liberi, dato che l'energia di ionizzazione dell'ossido è superiore a quella del silicio;
- le cariche generate restano intrappolate all'interno della well, causando lo shifting persistente delle tensioni.

L'accumulo di cariche è particolarmente grave perché il body dei MOSFET SOI-PD è flottante, ma può essere mitigato inserendo dei pozzi che colleghino il body al riferimento a massa. Questa contromisura si paga in termini di spreco di area e di fasi aggiuntive di processo, oltre a limitare alcuni dei vantaggi del SOI.

Per quando concerne i SEE, la tecnologia SOI è estremamente efficace rispetto alle alternative bulk. Oltre alla già citata immunità ai SEL, grazie alla ridotta quantità di portatori liberati i dispositivi basati su SOI presentano una resistenza ai SEU e ai SET superiore, nonostante presentino una carica critica minore.

Oltre alla maggiore sensibilità ai TID, i dispositivi SOI soffrono di un costo di produzione maggiore, legato soprattutto all'impiego di metodologie per la deposizione dello strato di ossido non comuni nell'elettronica commerciale.

Strati epitassiali e strato nascosto

Una tecnica che richiede minori variazioni dai processi commerciali standard è l'inserimento di uno strato epitassiale sotto alle well dei CMOS.

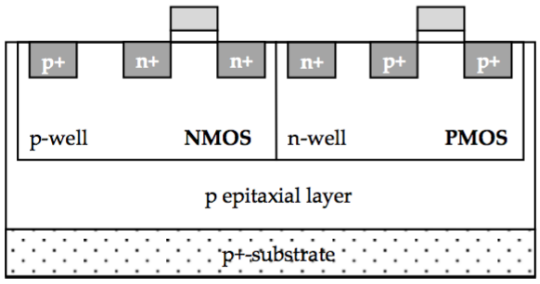


Figura 2.20: Epitaxial layer. Tratto da [1].

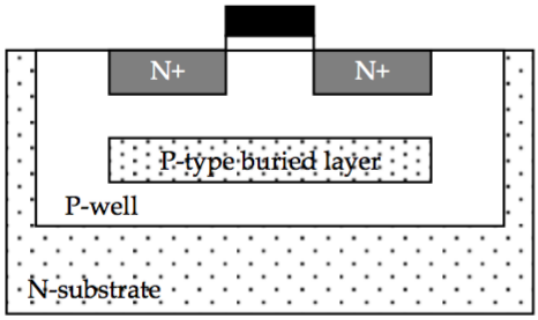


Figura 2.21: Strato sepolto. Tratto da [1].

Lo strato epitassiale cresciuto presenta una resistenza inferiore rispetto al silicio soprastante e riduce la catena di retroazione nei fenomeni di latchup fornendo un percorso alternativo alla corrente. Un effetto simile è ottenuto con lo strato sepolto, in cui la presenza di un picco di concentrazione di droganti in profondità causa la presenza di un campo elettrico che allontana le cariche libere dalla zona attiva. Entrambe le tecniche incrementano fortemente la carica critica e riducono il rischio di SEL.

Well Tripla

Anche in assenza di tecnologie SOI, le well possono essere isolate mediante l'impiego di giunzioni in inversa. Prendendo per ipotesi un substrato di tipo P, un PMOS in una N well è isolato da resto del dispositivo (se polarizzato correttamente). Per separare dal bulk un transistore NMOS, che richiede una well di tipo P, è necessario inserire uno strato di tipo N che si interponga fa la well e il substrato. Questa tecnica è detta well tripla, dato che la well P del NMOS è contenuta in una well di tipo N.

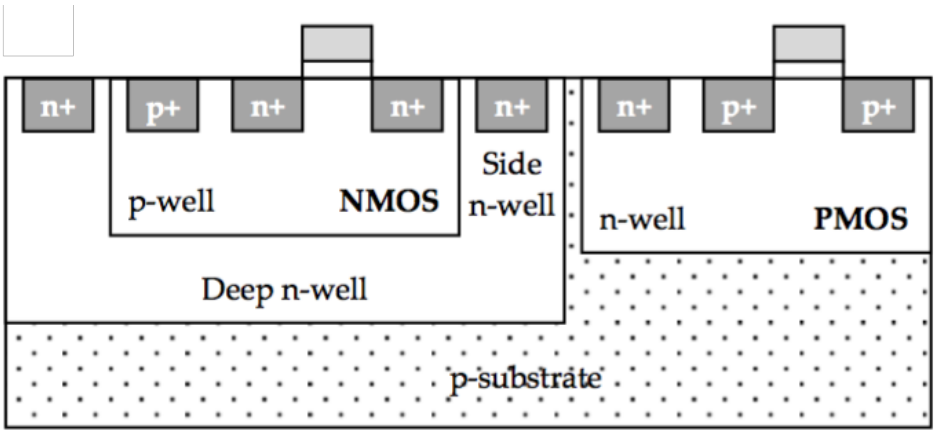


Figura 2.22: Tripla well. Lo strato sottostante è anche detto deep weel, mentre quello che separa il componente lateralmente è chiamato side well. Tratto da [1].

Per la sua struttura, la well tripla è a sua volta sensibile alla ionizzazione da radiazioni. Questa sensibilità è dovuta alla possibile presenza di foto corrente attraverso le due giunzioni. Sperimentalmente è stato osservato un incremento della sensibilità a certe radiazioni anziché una riduzione. Un ulteriore caveat è legato alle esigenze costruttive: per formare la deep weel (la parte inferiore della well) è necessario impiegare molteplici drogaggi che vanno a peggiorare la qualità del semiconduttore nella zona attiva ed ad incrementare la resistività nel canale. Di conseguenza l'applicazione di questa tecnica richiede una particolare attenzione durante la progettazione per dispositivi radiation hard.

Scaling e sensibilità

Non è semplice identificare una correlazione univoca fra scaling dei dispositivi e sensibilità alle radiazioni. L'analisi è molto complessa perchè l'evoluzione della costruzione dei dispositivi integrati è strettamente intrecciata allo sviluppo di processi e di materiali innovativi i cui effetti sulla radiation hardness sono profondi ma difficili da prevedere, come il già citato (e deleterio) impiego della passivazione BSPG[8].

Un altro parametro da tenere in considerazione è lo scaling delle tensioni. Con la riduzione delle tensioni di alimentazione, l'ampiezza del logic swing nei dispositivi digitali e del range nei circuiti analogici si sono ridotti, aumentando la sensibilità ai disturbi.

Se da un lato la riduzione delle dimensioni implica una zona sensibile più piccola per ogni transistore (o per ogni generico elemento di

memoria), la conseguente diminuizione delle capacità parassite (che in se è senza dubbio un pregio) è legata alla riduzione della carica critica e quindi una maggiore sensibilità sia alle cariche intrappolate nell'ossido sia a quelle generate durante gli eventi transitori.

Per quanto concerne la TID, con il passare da una generazione alla successiva la resistenza agli effetti cumulativi è calata[34]. Dal punto di vista storico questo trend ha costretto all'impiego di materiali a bassa radioattività nei package[27].

Gli effetti dei SEE in funzione della geometria e delle dimensioni non sono facili da valutare, quindi è difficile indicare un andamento preciso[35]. In generale, è possibile dire che anche per i SEE la sensibilità è aumentata. In ambito analogico questo effetto è legato alla riduzione delle capacità parassite, accompagnato da una riduzione della carica critica. Per i dispositivi digitali come CPU ed FPGA è interessante notare che la sezione d'urto per i SET aumenta anche con la frequenza di clock, oltre che con la densità d'integrazione e la diminuizione dello swing delle tensioni d'alimentazione[1].

2.5.3 Layout

Una volta scelta una tecnologia, anche non nata come radiation hardened, la resistenza alle radiazioni può essere incrementata con un layout dedicato.

Un approccio banale per ridurre il rischio di latchup in tecnologie bulk è allontanare MOSFET fra di loro, in modo da rendere la base dei BJT parassiti troppo lunga da causare la retroazione positiva. Lo svantaggio di questa tecnica è che limita fortemente la densità massima ottenibile.

Anelli di guardia

Il percorso parassita responsabile del latchup in una struttura CMOS si sviluppa vicino alla superficie. Un modo per ridurre la corrente che vi scorre in presenza di un SEL è porre un contatto in inversa che scarichi una parte della corrente di base all'esterno. La giunzione si può trovare in diretta solo in presenza di un latchup, ma non ha effetto durante il funzionamento regolare della struttura CMOS.

L'anello di guardia attorno al MOSFET può essere descritto come un secondo collettore del BJT parassita, che di conseguenza presenta un'amplificazione attenuata e necessita una carica critica molto maggiore per causare un SEL.

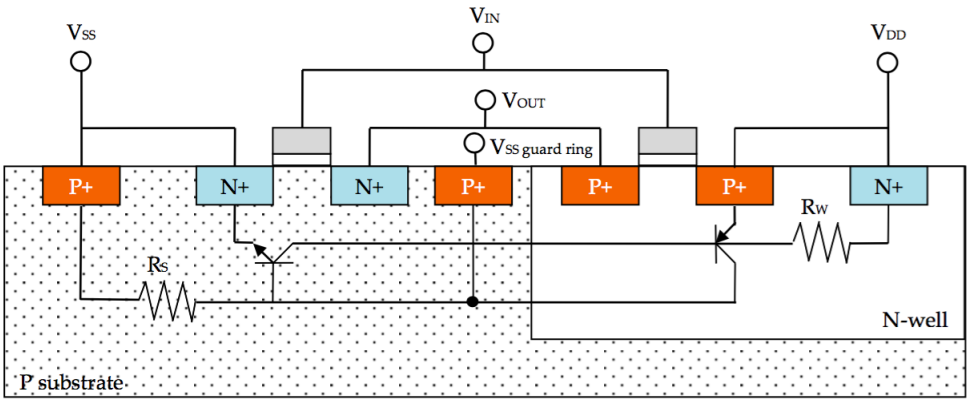


Figura 2.23: Guard ring

Enclosed Layout Transistor

Il design classico di un MOSFET può essere modificato in modo da abbattere la corrente perdita e irribustire il dispositivo verso i danni da TID. Una variazione particolarmente interessante è il Enclosed Layout Transistor (ELT), chiamato in letteratura anche Edgeless Transistor.

Esistono anche altre possibilità che garantiscono un minore spreco di area, come il Ringed Source Transistor e Ringed Inter Digitated Transistor. Tuttavia questi layout alterativi sono meno resistenti alla TID e più propensi a presentare una corrente di perdita rispetto agli ELT.

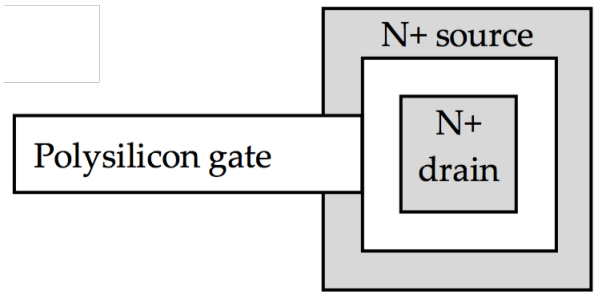
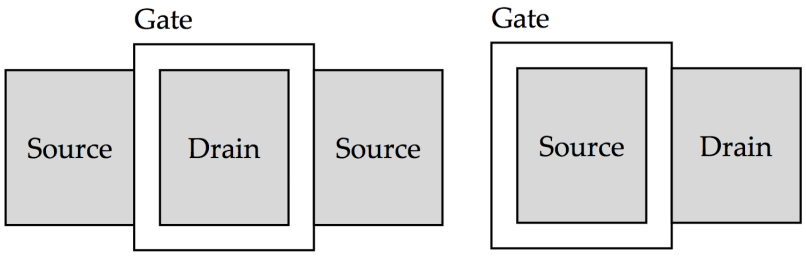


Figura 2.24: Enclosed transistor

L'overhead complessivo può essere trascurabile nel caso di dispositivi analogici, che presentano per loro stessa natura delle dimensioni maggiori rispetto a quelli digitali.



(a) Ringed inter digitated (b) Ringed source

Librerie radiazion hard

Lo sviluppo di una tecnologia radiation hard richiede ingenti costi di sviluppo che non possono essere ammortizzati con la produzione di grandi volumi, dato che le applicazioni che richiedono una elevata resistenza alle radiazioni sono limitate a pochi ambiti di nicchia. Un buon compromesso è di impiegare una tecnologia commerciale e irrobustirla mediante un layout specifico. Per questo motivo sono state sviluppate librerie di celle radiation hardened, comprendenti sia ELT e anelli di guardia sia componenti di livello più alto, come porte logiche, PLL e interfacce IO.

Come intuibile, l'impiego di componenti radiation hard comporta un overhead di area, che nelle librerie sviluppate per esempio da CERN e NASA si aggira attorno al 20%-40%[1].

2.5.4 Circuiti analogici e mixed signal

In un circuito digitale, il passaggio di una particella può indurre un SET che si può propagare lungo una catena logica o un SEU all'interno di una cella di memoria. Questo causa un errore su uno o più bit, che viene conteggiato nella BER o nella SER, a seconda del constesto. L'affidabilità è funzione di questi parametri.

In ambito analogico invece non esiste una metrica standard e la tolleranza ai SET deve essere valutata caso per caso. Per evidenziare la differenza in letteratura si impiega l'acronimo ASET, ovvero Analogic SET[36].

Per ridurre la carica raccolta in presenza di un singolo evento si possono impiegare tecnologie specifiche già presentate, come strati epitassiali, isolamento in inversa e anelli di guardia[39][40].

L'approccio "a forza bruta" è aumentare la carica critica necessaria per causare un errore. Questa via può essere percorsa senza impiegare topologie o processi radiation hardened, ma ricorrendo a semplici variazioni di schemi commerciali[38][37], per esempio:

- aumento delle dimensioni dei transistor;
- aumento delle correnti di controllo;
- aumento delle tensioni di alimentazione;
- aumento delle capacità.

Va notato che i dipositivi analogici richiedono aree maggiori rispetto a quelli digitali già per altri motivi, come per limitare offset, rumore ed errori di mismatch. Di conseguenza, l'overhead in termini di area non è penalizzante come nei circuiti digitali[37].

Un'altra strada è reiettare il disturbo causato dall'ASET limitando la banda del circuito o impiegando una retroazione elevata, ma l'applicabilità di questo approccio deve essere valutata caso per caso.

Separazione dei nodi e ridondanza

Un componente integrato come un transistore o una resistenza con determinate caratteristiche (come resistenza, capacità o corrente di saturazione) può essere realizzato in almeno due modi. Si può realizzare un singolo esemplare delle dimensioni necessarie oppure si possono realizzare più esemplari identici in punti diversi della superficie del chip e ottenere le caratteristiche richieste combinandoli in parallelo o in serie. Quest'ultima modalità di layout è standard nella progettazione analogica: permette di abbattere errori di mismatch e aumenta la precisione dell'applicazione[37].

Tuttavia, realizzare un componente come parallelo di più oggetti elementari ha un grosso vantaggio ai fini della radiation hardening: se il passaggio di una particella causa uno stimolo in uno solo degli elementi che compongono in parallelo la risposta in uscita dal sistema dovuta alla ionizzazione sarà mediata fra tutti i rami e quindi attenuata.

Notare che se gli oggetti elementari sono fisicamente lontani sul layout la probabilità che ricevano carica dallo stesso evento cala notevolemente. In altri termini, distanziare i rami diminuisce la carica raccolta ricevuta pur non abbattendo la carica critica, incrementando la resistenza ai SEE.

In un contesto di radiation hardenen electronics, è necessario rispettare una distanza minima nel layout anche fra i nodi del circuito. Alcune topologie di circuiti immuni a singoli SEE non funzionano se i nodi sono sottoposti a stimoli correlati, come può succedere se i transistori sono troppo vicini sul layout[41].

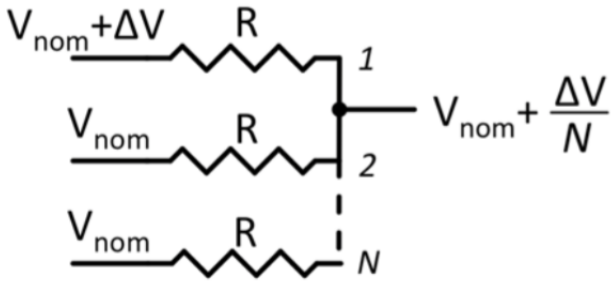


Figura 2.26: Un disturbo ΔV che colpisce un sistema composto da N resistori identici in parallelo viene ridotto a $\Delta V/N$.

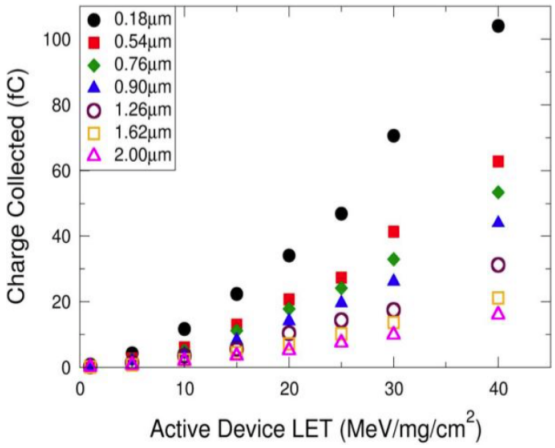


Figura 2.27: Quantità di carica in funzione della distanza fra PMOS. Misure effettute con stimolazione laser. Tratto da [41].

L'impiego di questa tecnica causa un aumento delle interconnessioni e della complessità del layout, con conseguente incremento dell'area occupata e riduzione della velocità.

Design differenzale

Alcune tipologie di layout mirano ad ottenere lo stesso disturbo su due linee diverse, per esempio cercando di minimizzare la distanza fra i drain dei due transistori. Anche il Common Centroid Layout, in cui si distribuiscono le varie copie del medesimo componente in parallelo in modo che il loro baricentro coincida, è una tecnica che può dare dei buoni risultati. In queste condizioni, la carica causa un disturbo a modo comune, che quindi può essere filtrato con un design differen-

ziale. Questo approccio è detto anche Differential Charge Cancellation (DCC).

Il vantaggio rispetto ad altre tecniche di radiation hardening è che permette una maggiore densità, anche se vi è un costo in termini di connessioni più complesse. Oltre a questo, vi sono i vantaggi classici del design differenziale, come una dinamica più estesa a parità di tensione di alimentazione e una maggiore resistenza al rumore.

Notare che anche nel caso peggiore, ovvero se la carica generata da un disturbo non si diffonde ad entrambi i transistori vicini ma colpisce solo una delle due linee, l'impatto sull'SNR è ridotto della metà.

2.5.5 Sistemi digitali

In ambito digitale, esiste un ampio spettro di schemi e variazioni per incrementare la resistenza alle radiazioni. La scelta è condizionata anche dalla disponibilità del mercato e sarebbe fuori luogo in questo contesto presentare una casistica dettagliata dei vari approcci, quindi verrà presentata solo una categorizzazione generale. Per prima cosa, nei sistemi digitali è necessario distinguere due differenti obiettivi.

Rilevazione dell'errore semplice: è il caso di reti con funzioni di controllo che rivelano un SET, ma non consentono di ottenere immediatamente il risultato valido.

Rilevazione e correzione dell'errore: è il caso di una rete che è capace di rilevare e correggere l'errore autonomamente.

In alcuni casi fortunati un controllo sulla validità del risultato richiede un overhead ridotto (per esempio il controllo del bit di parità), ma spesso non è così. L'approccio più diffuso è ripetere due volte il medesimo calcolo in due catene identiche e confrontare il risultato fornito. Se identico, entrambe le catene hanno funzionato correttamente o entrambe hanno commesso lo stesso errore. La seconda situazione è possibile ma improbabile. Per ridurre ulteriormente il rischio le due copie possono essere costruite in due zone del chip lontane fisicamente, eliminando la possibiltà di charge sharing.

Per ottenere la correzione dell'errore è necessario aggiungere almeno una ulteriore copia della medesima catena logica. In questo modo, se due copie su tre restituiscono il medesimo risultato allora è molto probabile che quello sia il dato corretto. Questa tecnica è detta Triple Modular Redundancy (TMR). Per ottenere una maggiore resistenza all'errore si può impiegare un numero maggiore di copie identiche, in questo caso si parla genericamente di N-Modular Redundancy (N-MR).

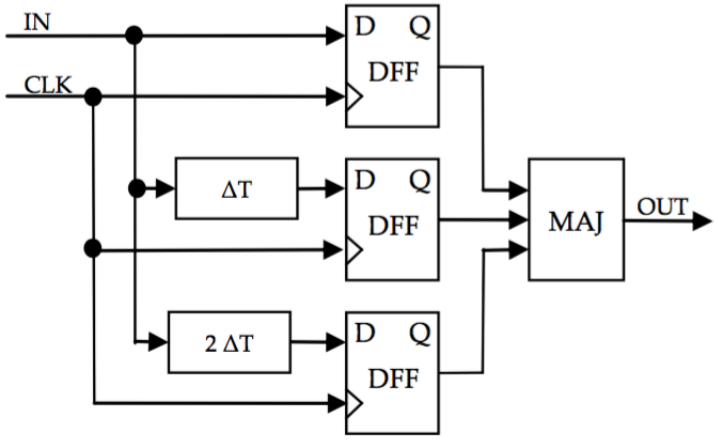


Figura 2.28: Esempio di ridondanza temporale con tre registri e un modulo di voto a maggioranza. Tratto da [1].

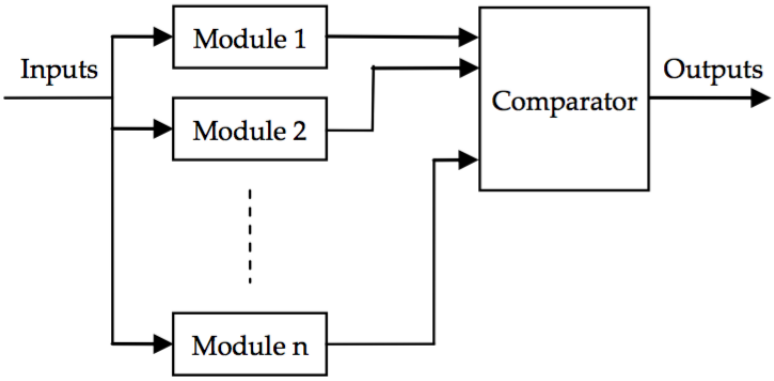


Figura 2.29: Ridondanza spaziale con N moduli. Tratto da [1].

Se la ridondanza è ottenuta ripetendo il medesimo calcolo su più reti logiche identiche si ha *ridondanza spaziale*, con un forte overhead in termini di area. La *ridondanza temporale* è invece ottenuta ripetendo in momenti differenti il medesiumo calcolo e confrontando i risultati ottenuti. Questa tecnica si paga sia in termini di riduzione della velocità massima raggiungibile sia di overhead di area, causato dalla presenza di elementi di memoria aggiuntivi. È importante notare che anche la circuiteria di controllo è a sua volta soggetta a SET e di conseguenza deve essere irrobustita a parte.

Anche per le memorie bisogna distinguere fra la rivelazione semplice e la correzione dell'errore. Per la rivelazione semplice, l'algoritmo più noto è il controllo di parità: si contano i bit pari ad uno e si rivela se

si è verificato un errore singolo. Per la correzione di un errore singolo, può bastare l'hamming code. La situazione si complica notevolemente se invece di un SEU singolo si hanno più errori sulla medesima parola (MBU). Per ovviare a questo inconveniente, sono stati sviluppati codici che possiedono una maggiore tolleranza all'errore.

Lo *scrambling* è una tecnica che consiste nel posizionare i bit della stessa parola distanti fra di loro. Un SEE che colpisca più elementi di memoria adiacenti ha una probabilità di causare un errore multiplo su una singola parola notevolmente più basso. In altre parole, lo scrambling fa in modo che l'errore venga distribuito su più parole (MCU) che sono facili da correggere singolarmente.

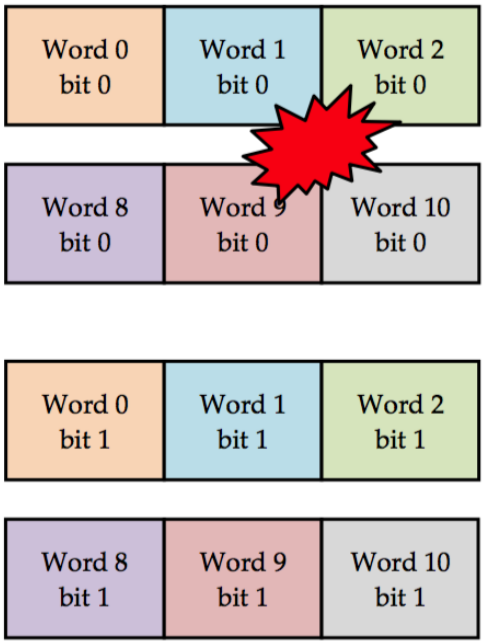


Figura 2.30: Scrambling. Tratto da [1]

Lo scrambling tuttavia comporta un routing più complesso e può limitare la velocità massima raggiungibile.

Sistemi programmabili

Nel caso particolare di microcontrollori e microprocessori progettati per applicazioni radiation hard spesso si usano tecniche di correzione dell'errore e di ridondanza a livello di registri, in modo trasparente al software.

Anche a livello di software si possono impiegare tecniche di incremento della radiation hardness. Si parla di Software Implemented hardware Fault Tolerance (SIFT), ovvero di sistemi resistenti agli errori grazie a tecniche software che possono essere implementate anche su tecnologie commerciali.

In questo caso si può solo impiegare tecniche di riduzione dei SET e SEU mediante ridondanza temporale, per esempio ripetendo due volte la stessa sequenza di istruzioni (ridondanza a livello di task) oppure anche una istruzione singola (ridondanza a livello di istruzione). Questo approccio, pur essendo limitato e molto costoso in termini di overhead, porta ad un forte risparmio in termini di progettazione e testing, che può essere interamente simulato con l'immissione di errori. In macchine dotate di sistema operativo la ridondanza può essere raggiunta anche a livello di applicazione.

Un caso particolare è il watchdog, una periferica che si trova frequentemente nei microcontrollori commerciali. Se il timer del watchdog non viene resettato regolarmente, dopo un numero noto di cicli di clock il sistema viene forzato ad uno stato iniziale noto (reset). Lo scopo del watchdog è di resettare il sistema nel caso in cui si verifichi un blocco del programma in esecuzione, come un SEFI. Il successo di questa tecnica è legato al basso overhead necessario, sia in termini di istruzioni sia di area occupata.

FPGA

Per le loro caratteristiche, le FPGA meritano un discorso a parte. Una prima distinzione ai fini della radiation hardness importante riguarda la programmazione:

- le FPGA basate su tecnologia antifuse sono OTP (One Time Programmable), una volta programmate la loro configurazione non può essere alterata;

- le altre FPGA conservano la propria configurazione all'interno una memoria SRAM o flash e di conseguenza può essere cambiata.

La prima categoria è intrinsecamente più resistente, dato che la programmazione viene memorizzata in maniera indelebile sulla struttura stessa dei circuiti. La tecnica è la stessa con cui si programma una PROM, bruciando alcuni percorsi dei circuiti interni e fissando alcuni nodi a livello alto o basso (a seconda della tecnologia), esattamente come succede in una ROM. Un SET può sempre verificarsi, ma non può corrompere in maniera permanente la programmazione.

Flash e SRAM invece sono soggette a SEU a causa delle radiazioni. Su un periodo lungo questi errori si accumulano fino a non essere più recuperabili e portare ad un SEFI grave. Per ripristinare la programmazione corretta si impiega lo *scrubbing*, ovvero si memorizza la configurazione su una memoria esterna e periodicamente si ricarica nella memoria della FPGA. È intuitivo che questa tecnica non permette di raggiungere l'affidabilità di una FPGA OTP e che causa un overhead.

Alcune FPGA radiation hardened sono progettate con celle logiche resistenti che implementano la TMR a livello locale oppure possono essere configurate per implementare un TMR a livello globale, mediante appositi algoritmi di place and route.

2.5.6 Mixed signal

Dato che i dispositivi mixed signal sono composti da una parte digitale e da una parte analogica, beneficiano delle principali tecniche di radiation hardening applicate sia in ambito digitale sia in ambito analogico. Esistono tuttavia alcune analisi specifiche della progettazione di circuiti misti.

2.5.7 Riduzione della finestra di sensibilità

Nel caso di dispositivi mixed signal come convertitori AD o DA e PLL esiste un segnale di clock periodico che determina la temporizzazione delle varie fasi di funzionamento del sistema. Durante alcune di queste fasi un disturbo causato da un evento esterno potrebbe propagarsi sull'uscita, mentre in altri momenti non avrebbe alcun effetto.

Per mostrare il concetto, prendiamo il caso semplice di un ADC ideale ad un 1 bit, composto da un blocco sample-and-hold seguito da un comparatore ed un flip flop D. Se si verifica un SEE che altera il valore campionato dal SH fra l'istante in cui il avviene il campionamento e l'istante in cui il valore viene letto dal registro, allora il valore sull'uscita è errato. Se si verifica in un altro momento, la carica liberata dal SEE non causa errore (almeno nel nostro modello ideale).

Si parla quindi di finestra di sensibilità del blocco SH, o Window of Vulnerability (WOV). Dato che si tratta di una frazione di un periodo ci si può riferire all'intervallo di fase e in questo caso si parla di Phase Dependent Sensitivity (PDS). Se la WOV è la metà del periodo, solo il 50% degli ASET causerà errore. Se può essere ridotta ulteriormente anche la sensibilità diminuirà dello stesso fattore.

In un caso reale, è necessario tenere conto del fatto che i vari nodi possono presentare finestre di sensibilità diverse. Nella pratica la

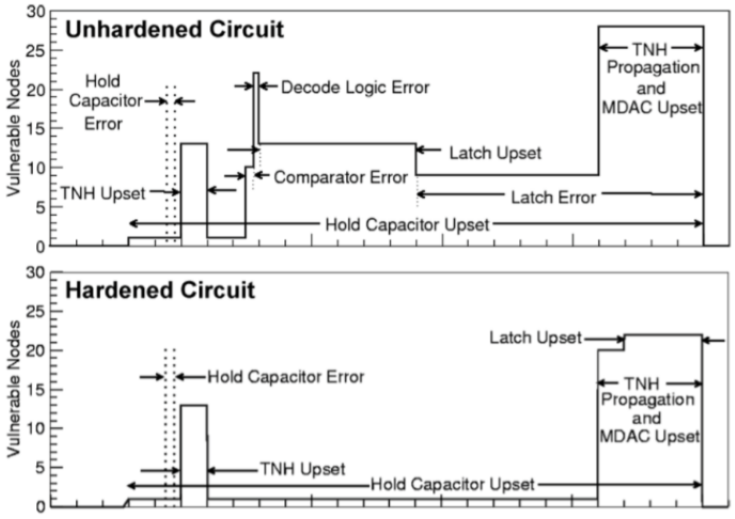


Figura 2.31: Esempio di finestra di vulnerabilità dei nodi di un convertitore ADC a 2 bit standard e radiation hardened. I tempi sono normalizzati rispetto al periodo. Tratto da [42].

riduzione della VOW può richiedere l'impego di topologie differenti o di un numero maggiori di fasi di clock, con conseguente aumento della complessità progettuale[42]. In aggiunta, questo tipo di analisi rischia di nascondere alcuni aspetti importanti ai fini della progettazione, in particolare il fatto che non tutti i nodi hanno lo stesso impatto sull'errore. Prendiamo ad esempio un generico ADC: un SEE che provochi un errore sulla valutazione del bit più significativo causerà una degradazione dell'SNR peggiore rispetto ad uno che colpisca un bit meno significativo.[43]

2.6 Test e simulazioni

Per una corretta valutazione del rischio è fondamentale poter effettuare test che simulino l'ambiente radioattivo in cui il dispositivo si troverà a funzionare, soprattutto perché l'affidabilità è un parametro importante in questi ambiti (in particolare militare e spaziale).

2.6.1 Collaudo in ambiente di lavoro

Il collaudo nell'ambiente finale presenta due grossi limiti:

- tempi di collaudo lunghi;
- costi elevati.

L'esposizione in ambiente reale è abbastanza bassa e per ricevere una dose sufficiente da valutare la resistenza alla dose sul lungo termine o un numero significativo di SEE serve molto tempo.

Si possono identificare almeno tre strategie principali per effettuare questo tipo di test.

Satelliti scientifici: alcuni progetti scientifici permettono di studiare il comportamento dell'hardware[44]. Esistono progetti mirati allo sviluppo di moduli standard a basso costo per esperimenti in orbita, come lo Space Environment Testbed (SET) sviluppato dalla NASA[45] o il più famoso CUBESAT[46].

Palloni stratosferici: i palloni sono capaci di portare carichi anche pesanti ad un costo relativamente basso. L'altitudine raggiungibile varia fra i 12 e i 45 km, per periodi molto lunghi[47].

Test a terra: i collaudi in volo presentano forti limiti in termini di peso, volume e potenza disponi. Un esempio emblematico è il test Rosetta, in cui sono stati impiegati cluster di 100 FPGA prodotte da Xilinix di varie famiglie[6], posizionate in 4 diverse locazioni e monitorate continuamente per mesi in cerca di SEU[48].

I test con radiazioni naturali sono molto lunghi, costosi e poco flessibili: quando possibile è preferibile effettuare altre tipologie di test.

2.6.2 Test accelerati a terra

Per sopperire alla bassa esposizione naturale e rendere più rapida la caratterizzazione dei dispositivi radiation hard una soluzione intuitiva è di impiegare sorgenti artificiali. La disponibilità e gli effetti dipendono dal tipo di radiazione.

Elettroni: prodotti in acceleratori per la valutazione della resistenza al TID.

Protoni: disponibili in grande quantità in acceleratori, hanno il difetto di causare anche danni da displacement.

Neutroni: dato che i neutroni mancano di carica, esistono meno acceleratori in cui è possibile ottenere fasci ad elevata intensità. Un'altra sorgente più reperibile per test preliminari sono gli isotopi radioattivi come il Californium-252.

[6] L'esperienza ha coinvolto un totale di 1700 esemplari di FPGA di due famiglie di dispositivi differenti (Spartan e Virtex), con chip con passo minimo di 90 nm, 130 nm e 150 nm.

Raggi α: prodotti con sorgenti radioattive come Californium-252 e Americium-241, sono impiegati principalmente per causare SEE sull'intero chip. Notare che, a causa della scarsa penetrazione, i test basati su particelle α non possono essere effettuati in aria ma solo in vuoto.

Raggi X: rispetto alle alternative sono i più economici e disponibili. Tuttavia la distribuzione delle cariche accumulate è poco uniforme.

Raggi γ: sono prodotti principalmente mediante isotopi radioattivi come Cesio 137 e soprattutto Cobalto 60. Un difetto sia dei raggi γ sia dei raggi X è lo scarso realismo: i fotoni sono una componente minoritara della TID in orbita.

Ioni pesanti: gli ioni pesanti sono impiegati per la caratterizzazione dei SEE.

Una nota a parte merita l'impiego di laser impulsati. Mentre le altre tecniche di test irraggiano l'intero chip, con l'impiego di un fascio laser localizzato è possibile eccitare con precisione elevata un punto specifico del chip. Gli impulsi possono essere temporizzati, permettendo di ricostruire situazioni difficili da osservare altrimenti[49][50].

2.6.3 Fault injection

Dato che allestire anche un test semplice è costoso e impegnativo, è di fondamentale importanza poter simulare l'avvenimento di un errore dovuto a radiazioni.

Sistemi digitali

Nell'elettronica digitale è una necessità comune poter simulare un errore e osservare come il sistema risponde. Di conseguenza esiste una ampia varietà di strumenti che permettono l'inserzione di un errore lungo una catena logica oppure il flipping di un bit in memoria. Il modello può essere anche a livello di gate. A livello superiore, un errore può essere introdotto all'interno di una CPU simulata, impiegando strumenti di debug software commerciali.

Tipicamente è possibile anche emulare il comportamento del sistema su un dispositivo reale, per esempio con l'impiego del JTAG. È intuibile i modelli impiegati sono ideali e di conseguenza non possono prescindere da analisi più approfondite a livello di singolo elemento di memoria o di cella logica.

Simulazione circuitale

A livello più basso, il passaggio di una particella è simulabile con l'impiego di generatori di corrente che riproducano un impulso di corrente fra i nodi del circuito. Questa tipologia di analisi circuitale è relativamente economica in termini di consumo di tempo e di risorse, ma rischia di nascondere interazioni dovute al layout e alla tecnologia.

Simulazione fisica

La simulazione fisica è necessaria, soprattutto se non esiste già una caratterizzazione empirica per una certa combinazione di layout e tecnologia.

Dato che i modelli impiegati richiedono una potenza computazionale superiore rispetto agli approcci precedenti, si cerca di ridurre e semplificarli il più possibile. Un primo passo è limitare la simulazione a pochi elementi vicini, al limite ad un solo modulo elementare come una cella di memoria o una porta logica. In secondo luogo, spesso è preferibile impiegare inizialmente modelli solo 2D e successivamente passare a modelli 3D[51].

Questa tesi si prefige lo studio di uno strumento flessibile per la simulazione fisica, che permetta lo studio della risposta in funzione delle tensioni e correnti del circuito ma anche dell'angolo di incidenza della particella e della sua energia, sia in 2D e 3D.

Capitolo 3

Strumenti disponibili

In questo capitolo verrà presentata una breve casistica degli strumenti in commercio. Successivamente verranno esposte le caratteristiche del linguaggio ad alto livello Python e i motivi che lo hanno reso popolare in ambito scientifico.

Infine saranno mostrate le caratteristiche di FEniCS, la piattaforma open source per il metodo agli elementi finiti che è stata impiegata in questo progetto.

3.1 Strumenti disponibili

Mentre la simulazione ad alto livello degli effetti dei SEE su dispositivi digitali può essere fatta con strumenti standard di iniezione degli errori e di debug, in ambito analogico la situazione è più variegata. Per quanto riguarda la simulazione a livello puramente circuitale è facile inserire all'interno di uno schema dei generatori di corrente che replichino gli effetti di un SEE e descrivere temporizzazioni anche complesse (come può essere la valutazione della finestra di vulnerabilità) con l'impiego di simulazioni miste analogico digitale. Il limite di questo approccio è che non può tenere conto di effetti di amplificazione della carica o di condivisione della carica dovuta a strutture parassite. Esistono modelli semplificati per descrivere questi fenomeni all'interno di una simulazione puramente circuitale, ma non sono affidabili quanto una simulazione fisica[26].

Per limitare il consumo di risorse, molti strumenti consentono solo una simulazione 2D oppure pseudo 3D, in cui sono eseguite svariate simulazioni solo 2D sul piano tangente alla superficie e solo successivamente sono unite lungo l'asse ortogonale[51].

Esistono due contesti principali in cui sono impiegati simulatori fisici di SEE. In ambito spaziale e militare l'attenzione è principalmente rivolta allo sviluppo di elettronica radiation hard, mentre nella ricerca esiste la necessità di valutare nel dettaglio gli effetti su sensori i cui materiali e strutture sono non standard.

3.1.1 Simulatori commerciali

In ambito commerciale esistono moduli integrabili all'interno di un flusso di progetto più ampio, come il REM (Radiation Effects Module) sviluppato da Silvaco[52] oppure la libreria per la simulazione di radiazioni ionizzanti per Sentaurus, sviluppato da Synopsys[53].

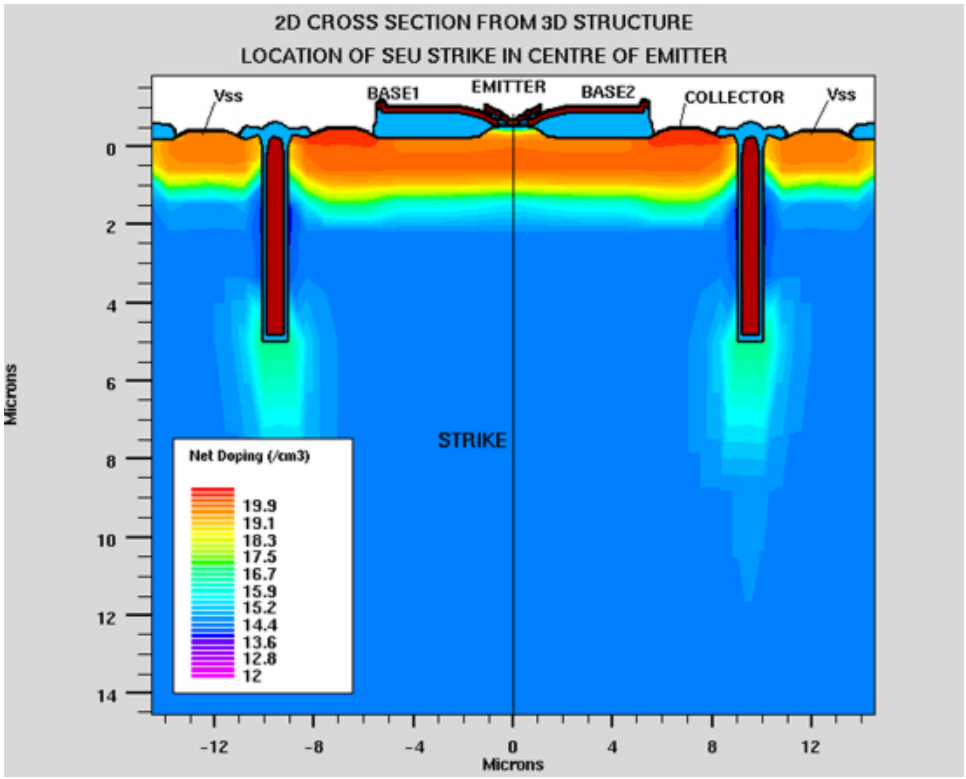


Figura 3.1: Simulazione di un SEU con il modulo REM di Silvaco. Tratto da [52]

Questi tools offrono la possibilità di simulare una ampia varietà di effetti oltre ai SEE, come i danni cumulativi causati da TNID e TID oppure come la produzione di fotocorrente. Tuttavia questi strumenti, pur offrendo un ampio ventagli di vantaggi, non possono essere modificati e sono vincolati al produttore.

3.1.2 Strumenti open source

Un approccio opposto è impiegato in ambito di ricerca: per esempio per valutare gli effetti del passaggio di particelle ionizzanti all'interno di sensori al silicio o al diamante. Queste tipologie di dispositivi hanno caratteristiche elettriche e geometriche speciali ed è complesso generalizzare queste applicazioni. Di conseguenza, il simulatore deve poter essere adattato secondo necessità. Come esempio, in questa sezione verrà analizzato brevemente un simulatore particolarmente completo, chiamato Weightfield 2, nato presso il Institute of High Energy Physics (HEPHY)[54] di Vienna e successivamente sviluppato presso l'INFN di Torino[55].

Il Weightfield è stato sviluppato per simulare il funzionamento di una classe particolare di sensori per il tracciamento delle particelle energetiche caratterizzata da una struttura composta da lunghe strisce parallele di metal adagiate su silicio drogato, con giunzioni fortemente polarizzate in inversa. Di fatto, l'intero sensore è un array di fotodiodi verticali completamente svuotati.

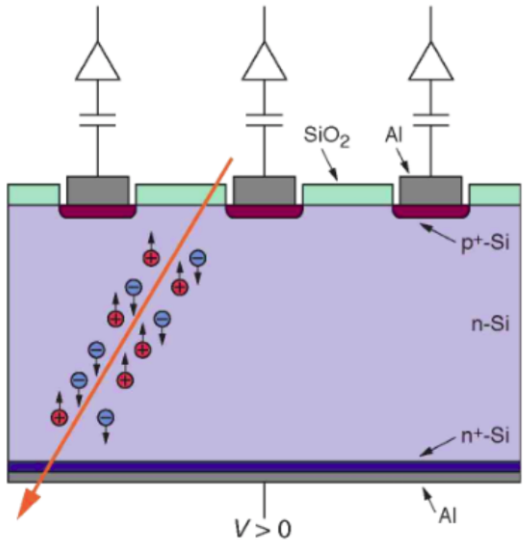


Figura 3.2: Struttura generale di un sensore a strip. Al passaggio di una particella una striscia catturerà la carica generata dall'evento, fornendo informazioni sulla traiettoria. Tratto da [55]

Al passaggio di una particella ionizzante, le strisce assorbono la carica generata e generano un impulso di corrente sull'uscita. La struttura è posta in condizioni di forte inversione, con tensioni nell'ordine delle decine di volt. Questo consente di semplificare l'algoritmo: a differen-

za di quanto accade con un modello drift-diffusion, in questo caso il contributo al campo elettrico dato dalle cariche generate è trascurabile. Il potenziale elettrostatico quindi è costante nel tempo. Esistono inoltre altre due ottimizzazioni possibili: la stuttura è simmetrica rispetto ad un asse e di conseguenza la simulazione può essere effettuata in 2D senza perdita di precisione ed è anche periodica lungo l'asse ortogonale. Di conseguenza, è sufficiente calcolare il campo generato da un solo elettrodo isolato e ricavare il campo elettrico complessivo per sovrapposizione degli effetti.

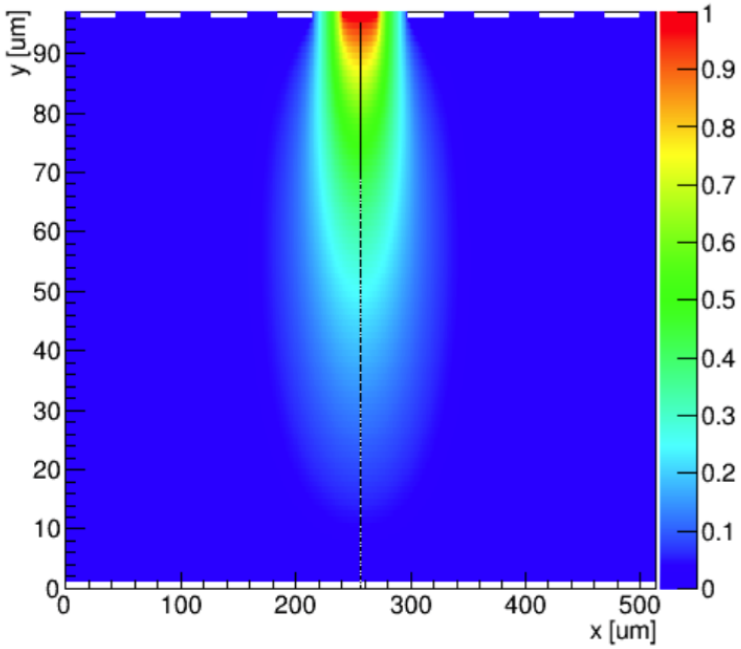


Figura 3.3: Simulazione di un una strip singola con Weightfield. Tratto da [55]

Come nota a margine, dato che il codice del programma è liberamente disponibile, possiamo notare che l'equazione di Poisson per il potenziale è risolto impiegando il metodo delle differenze finite su di una griglia regolare, mentre la propagazione dei portatori è simulata con una variante del Metodo Monte Carlo.

La possibilità di usare un motore per la simulazione modificabile ha permesso una intensa ottimizzazione, mentre questo non sarebbe stato possibile con un strumento commerciale chiuso. Naturalmente questa è una applicazione di nicchia e il fatto che sia modificabile ed adattabile a nuovi problemi non elimina il fatto che questa operazione sia complessa. In secondo luogo, questo tool si basa pesantemente sull'am-

biente ROOT[56], la cui diffusione è molto limitata fuori dall'ambito della fisica delle alte energie[1].

Da questa situazione è nata l'esigenza di uno strumento multipiattaforma, facile da configurare e che consentisse di adattare il simulatore a nuovi problemi rapidamente. Il primo passo nella progettazione è stato scegliere un linguaggio disponibile pressoché su ogni piattaforma di calcolo che non richiedesse molto tempo né per l'apprendimento né per lo sviluppo ma che al contempo possedesse un ampio spettro di librerie, scientifiche e non: Python.

3.2 Python

Python è un linguaggio di programmazione interpretato general purpose di alto livello che è diventato molto popolare in ambiente scientifico negli ultimi anni. Parte di questo successo è dovuto alla sintassi facile da comprendere, vicina allo pseudo codice: la filosofia di Python enfatizza la semplicità e la leggibilità del codice. Parte è legato allo stretto legame con linguaggi ad alta efficienza come C e C++: l'implementazione più diffusa è CPython, un interprete che implementa tutte le funzioni direttamente in codice C ottimizzato, limitando l'overhead dovuto all'uso di un linguaggio interpretato. La creazione di un modulo in C++ o in C da integrare all'interno di un progetto scritto in Python è una pratica consolidata. Esistono anche strumenti che permettono di automatizzare la procedura. Elenchiamo brevemente le caratteristiche peculiari del linguaggio.

Rinforzo delle convenzioni

Lo stile con il quale viene scritto il codice incide fortemente sulla sua leggibilità. La sintassi di Python incentiva l'uso delle buone norme di programmazione, prima fra tutte l'indentazione. L'uso dell'indentazione è consigliato ma facoltativo in altri linguaggi di ampia diffusione come C++ o Java, mentre in Python è intrinseca nella sintassi e viene impiegata per delimitare regioni di codice come funzioni e classi. L'uso delle parentesi graffe per delimitare i blocchi è rindondante e appesantisce la lettura, quindi in Python è stata abolito. Anche l'uso di parentesi tonde nel caso di le espressioni condizionali e di cicli non è necessario ed è stato eliminato.

[1] ROOT è un tool sviluppato presso il CERN che mette a disposizione molte funzionalità specifiche per il calcolo scientifico e consente l'integrazione di codice FORTRAN, Python e C++.

Documentazione

Come in C++, in Python è possibile definire funzioni, classi e librerie. Ad ognuno di questi oggetti viene implicitamente aggiunta una stringa scritta dallo sviluppatore, detta *docstring*. Questa stringa è accessibile dall'esterno dell'oggetto come gli altri attributi, che sono sempre pubblici a differenza di altri linguaggi come il C++ e il Java.

La distribuzione standard di Python include anche una utility chiamata *pydoc* che usando le docstring all'interno di un modulo genera automaticamente della documentazione in formato html, accessibile con qualsiasi browser. In alcuni progetti le docstring sono impiegate anche per contenere esempi per l'utente finale e test da eseguire per verificare il corretto funzionamento del codice. In questo modo eventuali incoerenze fra il comportamento reale del modulo e quello descritto nella documentazione vengono scoperte rapidamente.

Interfaccia interattiva

Qualsiasi distribuzione di Python (con l'ovvia eccezione dei sistemi embedded) offre un'interfaccia da terminale (CLI) per scrivere ed eseguire i comandi passo passo e osservare immediatamente il risultato dell'esecuzione, come è possibile fare nell'ambiente Matlab. In aggiunta, le docstring dei moduli sono accessibili sia direttamente come attributi dell'oggetto sia mediante una funzione help() inclusa di default nell'ambiente interattivo.

Esistono anche interfaccie interattive alternative che possono offrire funzionalità aggiuntive, come IPython[57] che offre un ambiente integrato con pacchetti impiegati in ambito scientifico, per esempio per analisi numeriche e visualizzazione di grafici. Questa shell offre anche delle funzionalità assenti nella versione di default estremamente utili (come l'autocompletamento) che sono presenti nelle CLI general purpose moderne.

Comandi immediatamente riconoscibili

In Python alcune operazioni semplici hanno una sintassi intuitiva. Per chiarire questo punto conviene vedere alcuni esempi. L'espressione logica

$0 < \mathrm{x} < 1$ **and** $0 < \mathrm{y} < 1$

ha significato universale ed univoco, ma è anche un'espressione valida in Python. In C/C++ sarebbe accettata[2] ma a causa dei casting automatici sarebbe interpretata come

```
0 > x and 0 > y
```

che è un palesemente errata. L'equivalente codice in C/C++ è meno leggibile e più facile da sbagliare:

```
(0 < x) && (x < 1) && (0 < y) && (y < 1)
```

Notare che anche quest'ultima notazione è accettata in Python. Un altro aspetto è la gestione automatica della memoria. Pur incidendo negativamente sulle performance, con questa tecnica si elimina la necessità di molti costrutti per la gestione degli oggetti dinamici (come malloc() e free() in C) ed si evitano errori nel loro utilizzo. Un esempio banale è la manipolazione delle stringhe. Per eseguire la concatenazione, ovvero la "somma" di due o più stringhe, è sufficiente un comando:

```
>>> 'Py' + 'th' + 'on'
'Python'
```

In altri linguaggi questa operazione non è disponibile nativamente, come in C, ma richiede delle manipolazioni e dei controlli in più che allungano il numero di linee di codice necessarie. Un altro esempio tipico è la gestione dei numeri. In C/C++ l'utente deve effetturare tutti i controlli necessari per evitare errori di rappresentazione. L'interprete Python gestisce numeri interi di qualsiasi dimensione indipendentemente dalle caratteristiche della piattaforma senza bisogno di comandi aggiuntivi.

Ampia disponibilità di librerie

Grazie alla facilità con cui si possono creare moduli per Python in C/C++, nel corso degli anni sono state create librerie per ogni tipo di applicazione. Esistono librerie per il trattamento di audio e video, per la creazione di videogame, per la risoluzione di espressioni algebriche, per l'intelligenza artificiale o anche per il riconoscimento vocale.

Anche senza usare librerie esterne, il paccheto Python comprende un ampio parco di funzioni, per esempio:

Funzioni numeriche Numeri complessi, frazioni, generazione di numeri casuali

[2] L'uso di operatori 'and', 'or' e 'not' è possibile anche in C++, ma raro. In C non è standard.

Gestione file Operazioni complesse come copia e spostamento

Database Interfaccia con database come squlite3 e pickle, uno standard specifico per Python[3]

File compressi Supporto di file compressi come .zip e .tar

Multitastking Creazione di thread e processi e gestione di strutture condivise

Email Ricezione e invio di email

Linguaggi di markup Creazione e lettura di file html e xml

Web Controllo delle connessioni, elaborazione di pagine web e anche gestione di server

Strumenti di sviluppo Debug e testing dell'esecuzione di programmi Python

Crittografia Gestione di hashing di messaggi e password

Interfaccia grafica Creazione di una interfaccia grafica basata su Tk

Per impiegare librerie esterne esistono gestori di pacchetti che facilitano le operazioni di installazione e aggiornamento. Quello di default è pip, disponibile sia in ambiente Unix sia in ambiente Windows. Il suo utilizzo ricalca quello di gestori di pacchetti generici più famosi come apt-get o yum.

3.3 Scipy e Numpy

In ambiente scientifico la libreria di Python più nota e completa è Scipy. Scipy fornisce una serie di strumenti per l'elaborazione numerica, dall'acquisizione di dati fino alla visualizzazione di grafici in 3D. Il pacchetto è separato in moduli:

- NumPy
- Libreria di Scipy
- Matplotlib

[3] Il modulo pickle permette di salvare su file qualsiasi oggetto Python, comprese funzioni, classi e anche l'intero contesto dell'esecuzione.

- Pandas
- SymPy
- IPython
- Nose

3.3.1 Numpy

Il modulo centrale dell'intero progetto è Numpy, che fornisce un framework per la manipolazione di vettori e matrici multidimensionali. Dato che in applicazioni reali il numero elementi può essere molto grande, è fondamentale impiegare delle tecniche specifiche per ridurre sia la memoria occupata sia il carico computazionale.

Array Vettori e matrici possono essere implementati come liste dinamiche, ma questo porta ad un forte overhead. In Numpy vettori e matrici sono degli array con un numero fisso di elementi con ordinamento variabile. In questo modo si può sfruttare la contiguità degli elementi in memoria per accelerare le operazioni di lettura e scrittura. È possibile scegliere anche il tipo di dato, in modo da ridurre lo spazio occupato al minimo necessario per l'applicazione o per aumentare la precisione. È possibile usare dati booleani, interi con segno e senza segno, stringhe e numeri in virgola mobile, che possono essere impiegati anche come numeri complessi. La precisione massima per i numeri in virgola mobile è di 128 bit, ma esistono tecniche per superare questo limite. Notare che Numpy è un framework per vettori e matrici densi, in cui gli elementi nulli sono pochi. È inadatto al trattamento di matrici sparse, in cui solo una piccola parte degli elementi è diversa da zero.

View Manipolare una matrice non è particolarmente oneroso, ma lo è scrivere una nuova matrice in memoria. Per questo motivo conviene limitare la scrittura e la lettura dei dati alle operazioni che lo richiedono espressamente. Per le operazioni che invece si limitano a modificare le coordinate degli elementi senza variarne il valore è opportuno impiegare una struttura dati specifica, denominata view. La view permette di "vedere" la matrice in modo diverso, alterando soltanto la modalità di accesso ai suoi elementi. Prendiamo in considerazione la trasposizione di una matrice:

```
>>> a = numpy.array([[0,1,2], [3,4,5], [6,7,8]])
>>> a
```

```
array([[0,  1,  2],
       [3,  4,  5],
       [6,  7,  8]])
>>> b = a.transpose()
>>> b
array([[ 0,  3,  6],
       [ 1,  4,  7],
       [ 2,  5,  8]])
```

La matrice b è la view della trasposta di a. La creazione della matrice b è un'operazione veloce, dato che non ha richiesto scritture o letture in memoria. La view si può usare per leggere e modificare la matrice originale accedendoci come se fosse trasposta:

```
>>> b[1][0] = 10
>>> b[2][0] = 10
>>> a
array([[ 0, 10, 10],
       [ 3,  4,  5],
       [ 6,  7,  8]])
```

Le view sono impiegate anche per cambiare le dimensioni o il tipo dei suoi elementi una matrice, per esempio una matrice 10x10 di numeri complessi può essere "vista" come una matrice 10x20 o 20x10 di numeri in virgola mobile.

Per incrementare le prestazioni, Numpy si appoggia a librerie esterne come LAPACK e ATLAS, impiegate anche da altri framework per il calcolo numerico come Matlab e Octave. Queste librerie tuttavia richiedono un compilatore Fortran (preferibilmente g77 o gfortran). Per rendere il pacchetto più flessibile, Numpy è dotato di implementazioni delle medesime funzioni in C da usare nel caso in cui il compilatore manchi o non sia compatibile con la piattaforma in uso.

3.3.2 Libreria di Scipy

All'interno di Scipy coesistono una serie di strumenti eterogenei basati su Numpy, raccolti sotto il nome di Libreria di Scipy. La libreria è formalmente divisa in sezioni in base agli scopi:

Funzionalità di base Costanti matematiche e fisiche con relative unità di misura

Funzioni speciali (scipy.special) In questa classe rientrano funzioni di interesse matematico come le funzioni di Bessel e di Airy di vario ordine

Integrazione (scipy.integrate) Implementazioni di algoritmi di integrazione, per esempio quadratica e trapeziodale

Ottimizzazione (scipy.optimize) Ricerca del minimo o del massimo con vari algoritmi

Interpolazione (scipy.interpolate) Algoritmi di interpolazione di vario ordine in una o più dimensioni

Trasformata di Fourier (scipy.fftpack) FFT n-dimensionale e calcolo della convoluzione di trasformate di Fourier

Signal Processing (scipy.signal) Filtraggio di segnali

Algebra lineare (scipy.linalg) Risoluzione di sistemi lineari, calcolo del determinante, dell'inversa, degli autovalori e autovettori di una matrice, decomposizione LU e di Cholesky e interpolazione ai minimi quadrati

Routines di gestione di grafi compressi sparsi (scipy.sparse.csgraph)

Strutture dati per lo spazio (scipy.spatial) Algoritmi per la suddivisione dello spazio in sottodomini

Statistica (scipy.stats) Calcolo di variabili aleatorie di varia distribuzione

Processing di immagini (scipy.ndimage) Filtraggio e confronto di immagini

File IO (scipy.io) Lettura e scrittura di file di vari standard, come Matlab, wav e testuali

Weave (scipy.weave) Integrazione di C/C++ inline

L'aspetto interessante del pacchetto Weave (disponibile anche standalone) è che permette di integrare blocchi di codice in C/C++ all'interno di un programma Python con estrema facilità. Il codice integrato è contenuto in una stringa e viene compilato solo quando necessario (Just-in-Time) e poi eseguito. Esistono due modi per impiegare queste funzionalità. A scopo dimostrativo vediamo il più semplice. Definiamo tre matrici 3x3 i cui elementi sono tutti pari a 1, chiamate a, b e c.

```
>>> a = numpy.ones((3,3))
>>> b = numpy.ones((3,3))
>>> c = numpy.ones((3,3))
```

Sommiamo a e b e salviamo il risultato nella matrice c. Per ottenere questa operazione Numpy offre una funzione detta vectorize che consiste in un ciclo che scorre tutti gli elementi delle matrici e li calcola uno per uno in Python. Con Weave si può eseguire la stessa operazione molto più rapidamente in C.

```
>>> expression = "c=a+b"
>>> scipy.weave.blitz(expression)
>>> c
array([[ 2.,  2.,  2.],
       [ 2.,  2.,  2.],
       [ 2.,  2.,  2.]])
```

La prima volta che viene chiamata la funzione blitz l'espressione viene compilata, ma le volte successive sarà eseguita immediatamente a partire dal codice già compilato, ottenendo velocità fino a 30 volte superiori rispetto a quanto permesso da Numpy.

In aggiunta, la libreria di Scipy offre il supporto per le matrici sparse, cioè matrici in cui solo pochi elementi sono diversi da zero. Questo tipo di matrici è tipico dei sistemi di equazioni differenziali discretizzate.

3.3.3 Altre componenti

Matplotlib è un modulo che offre le funzionalità di base per visualizzare grafici in 1D, in 2D e in 3D. Anche questo modulo dipende da Numpy. La sintassi impiegata in Matplotlib ricalca quella di Matlab, in modo da rendere più semplice l'apprendimento per gli utenti che hanno già esperienza con questa piattaforma.

Pandas è impiegato per l'acquisizione e l'elaborazione di grandi quantità di dati, che possono essere facilmente filtrati o ordinati impiegando dei comandi specifici. Pandas può accedere a database SQL e file in formato testuale, HDF5, Microsoft Excel e CSV, gestendo senza problemi sia dati mancanti sia ordinamenti personalizzati.

Per facilitare l'uso in ambiente scientifico, Scipy include anche una interfaccia a linea di comando simile all'interprete interattivo di Python, ma con una sintassi estesa e delle funzionalità aggiuntive come l'autocompletamento: il già citato IPython.

Scipy consente di eseguire operazioni simboliche, attraverso il pacchetto SymPy. Le operazioni possibili spaziano dalla semplificazioni di polinomi ed espressioni trigonometriche fino alla risoluzioni di limiti,

dal calcolo dimensionale alla descrizione di sistemi quantistici. SymPy dispone anche di una interfaccia apposita per la scrittura di espressioni matematiche per IPython, che permette di visualizzarle su terminale come si potrebbe fare con un visualizzatore Latex.

Infine all'interno di Scipy è presente anche un framework per la valutazione delle performance e il debug, detto Nose.

3.4 FEniCS

Storicamente, la complessità nella risoluzione di sistemi di equazioni differenziali ha causato il proliferare di tante librerie software per la risoluzione di problemi specifici, sviluppate con lo scopo di ottenere la massima velocità nella risoluzione attraverso un'elevata specializzazione. Tuttavia, questa idea porta anche degli svantaggi gravi. L'uso di codice specializzato a basso livello richiede un elevato grado di esperienza, oltre ad essere dissipativo in termini di tempo e di lavoro e foriero di errori difficili da individuare.

Un approccio innovativo è usare codice altamente specifico in modo da ottenere elevate prestazioni generato per via automatica partendo da una definizione ad alto livello del problema, riducendo sia il rischio di errori sia il tempo necessario. Questa è l'idea che ha ispirato FEniCS.

FEniCS è una suite di strumenti open source per l'automatizzazione dell'intera catena di operazioni necessarie alla risoluzione di equazioni differenziali con la tecnica degli elementi finiti, dalla definizione del problema fino alla visualizzazione.

Vediamo brevemente come funziona la generazione con FEniCS. L'utente definisce la forma variazionale del problema, compreso lo spazio di funzioni. Questa parte viene compilata in un programma specifico che viene eseguito usando i dati del problema forniti dall'utente come le condizioni al contorno o il mesh, che possono essere letti da un file in formato standard. Tuttavia, se il problema lo permette, parte della generazione dei dati può essere affidata al codice specializzato compilato sul momento, come visto con il pacchetto Weave.

Esistono altri tool software che permettono di automatizzare parzialmente la generazione di codice per FEM compilato ad alta efficienza, come FreeFEM++, Sundance, GetDP e LifeV. FEniCS tuttavia è quello che automatizza una porzione più ampia del processo e che rendere indipendenti la definizione del problema variazionale e della base di funzioni campione, rendendo di fatto più facile il supporto di una ampio spettro di campione. Per dare un'idea, FEniCS supporta nativamente le seguenti funzioni campione (in ordine alfabetico):

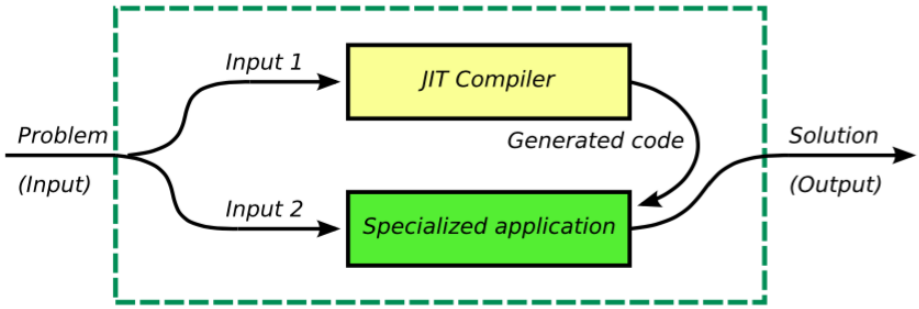


Figura 3.4: Il sistema usa un compilatore Just-In-Time per genereare del codice specifico per il problema in ingresso. Solitamente Input 1 è il problema variazionale insieme allo spazio di funzioni e Input 2 è composto dal mesh e dalle condizioni al contorno. Tratto da [62]

- Argyris
- Arnold-Winther
- Brezzi-Douglas-Fortin-Marini
- Brezzi-Douglas-Marini
- Bubble
- Crouzeix-Raviart
- Discontinuous Lagrange
- Discontinuous Raviart-Thomas
- Hermite
- Lagrange
- Mardal-Tai-Winther
- Morley
- Nedelec
- Quadrature
- Raviart-Thomas
- Real

Tutti questi spazi di funzioni sono disponibili con un numero di ordine arbitrario[4] e possono essere uniti per ottenere spazi di funzioni misti. Se necessario, un utente può comunque definire un altro spazio di funzioni da usare ed impiegarlo nella risoluzione di un problema senza avere necessità di modificare la definizione del problema variazionale.

L'utente finale può accedere alle funzionalità offerte da FEniCS ricorrendo a scripts scritti in C++ o in Python (o usando la shell interattiva di quest'ultimo, naturalmente). Le due interfaccie sono pressoché equivalenti sia sotto il profilo delle prestazioni sia della potenza espressiva. La libreria in Python è composta da wrapper delle funzioni e oggetti della libreria in C++. Una porzione predominante degli oggetti in Python è generata per via automatica usando SWIG[68][69], uno strumento specifico per la generazione automatica di codice di alto livello (come Python, Ruby, Java) a partire dalle API scritte in C/C++. Solo per una parte piccola è stato necessario adattare il codice a mano, per sfruttare le caratteristiche del linguaggio.

3.4.1 Storia e struttura

Nel 2002 nasce DOLFIN, che costituisce il nucleo di FEniCS. Lo scopo originario di questo progetto sviluppato presso il Chalmers University of Technology era di provvedere una libreria open source per la risoluzione di equazioni differenziali che sostituisse il framework Diffpack, che essendo un pacchetto proprietario poneva dei seri limiti alla modifica del codice sorgente. DOLFIN è la base su cui nasce FEniCS l'anno seguente grazie alla collaborazione dell'Università di Chicago, a cui si sono aggiunti successivamente il Argonne National Laboratory, l'Università di Delft, il Royal Institute of Technology KTH, il Simula Research Laboratory, la Texas Tech University e l'Università di Cambridge.

DOLFIN era una libreria monolitica in C++ che comprendeva l'implementazione di funzionalità per la definizione e la risoluzione di problemi lineari, oltre alla generazione di griglie (*mesh*) e di basi di funzioni o la definizione di problemi variazionali.

Durante lo sviluppo, queste funzionalità si sono espanse e sono state delegate a nuovi moduli che compongono FEniCS o a librerie esterne di terze parti. Anche se la modularità comporta il degrado delle prestazioni a causa dell'overhead, esistono diversi vantaggi.

[4] Ovviamente l'ordine scelto deve avere senso in termini matematici: per esempio se si cercasse di definire uno spazio campione di "numeri reali del secondo ordine" con FEniCS si otterrebbe un errore, dato che non esiste una definizione standard di questo tipo.

Debug Moduli indipententi possono essere testati separatamente, così da ridurre l'estensione del codice da controllare e semplificare la ricerca degli errori, inevitabili in un progetto così ampio.

Velocità dello sviluppo Un set più piccolo di funzioni è più semplice da implementare e ottimizzare, mentre la creazione e il mantenimento dei vari moduli può essere fatta in parallelo.

Flessibilità Un modulo creato ad hoc dall'utente finale o da terze parti può essere integrato per aggiungere funzionalità specifiche utili per la definizione di un problema o che permettano di ottimizzare il carico computazionale sfruttando caratteristiche specifiche del problema o della piattaforma. Un esempio interessante è la possibilità di sfruttare il parallelismo permesso dalla GPU mediante una interfaccia apposita basata su CUDA[64].

Vediamo brevemente quali sono i moduli standard in FEniCS

DOLFIN

DOLFIN è la libreria C++/Python che, come detto in precedenza, include la maggior parte delle funzioni necessarie al funzionamento di FEniCS. Per offrire un'interfaccia all'utente semplice da utilizzare e coerente, DOLFIN fa da tramite sia con i moduli interni sia con eventuali software esterni. L'utente può definire la geometria del sistema o i problemi variazionali senza dover mai interagire con un ambiente diverso da quello diverso da DOLFIN, sia in Python sia in C++.

UFL

Il framework usato i problemi variazionali è chiamato UFL (Unified Form Language). Per facilitare la traduzione di un problema matematico in UFL quest'ultimo è progettato per essere equivalente ai termini matematici che lo descrivono. Sia lo spazio delle funzioni sia le espressione in forma debole necessarie a caratterizzare il problema sono espressi con l'UFL.

UFC

L'UFC (Unified Form-assembly Code) è il framework usato per la definizione dellla discretizzazione agli elementi finiti del problema variazionale. Consiste in una interfaccia standard che comunica con le funzioni a livello inferiore per la costruzione e la valutazione delle matrici necessarie per la soluzione del problema. Di fatto, l'UFC componeva il

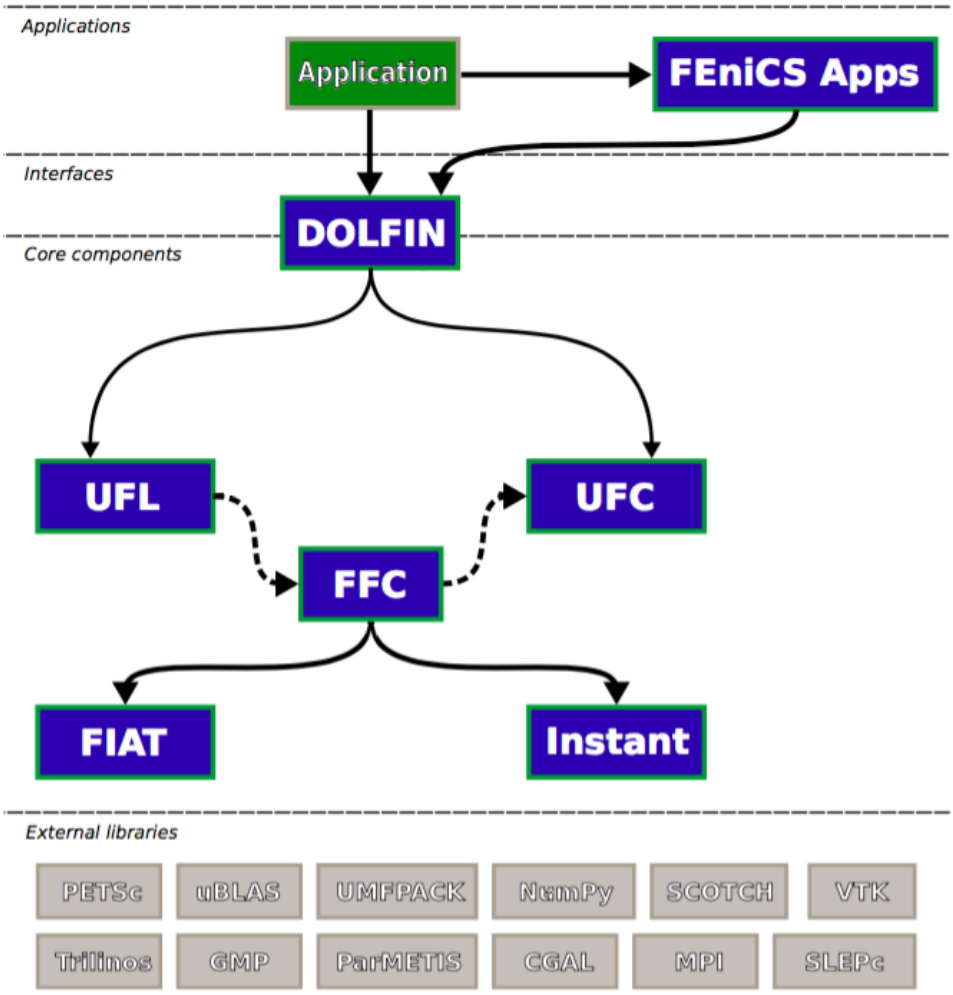


Figura 3.5: FEniCS è un progetto modulare. Adattato da [62]

layer fra l'FFC e il codice in C++, ma nelle ultime versioni di FEniCS (dalla 1.4) è stato incluso formalmente nell'FFC.

FFC

La generazione di codice di basso livello in C++ è una delle caratteristiche fondamendamentali di FEniCS. Il modulo che si occupa della compilazione a basso livello delle espressioni in cui sono espressi i problemi variazionali è il FFC (FEniCS Form Compiler). I problemi sono descritti utilizzando uno standard condiviso ad alto livello, ovvero l'UFL.

FIAT

La generazione degli spazi di funzioni incorporate in FEniCS (come Lagrange o Galerkin discontinuo di ordine arbitrario) è affidata al FIAT (FInite element Automatic Tabulator).

Instant

Instant è un modulo Python basato su SWIG e Distutils[70] usato per l'inlining di codice C e C++ all'interno di uno script Python. Instant è usato per la compilazione just-in-time (JIT) delle espressioni.

Moduli obsoleti

Originariamente, FEniCS includeva un modulo chiamato Viper dedicato per la visualizzazione immediata dei risultati basata su VTK, ma la funzione di plotter è stata ormai inclusa all'interno di DOLFIN. Altri moduli che non sono più sviluppati o che sono stati separati dal corpus del progetto sono ASCoT (usato per la valutazione della stabilità della discretizzazione impiegata in certe classi specifiche di problemi), SyFi/SFC (un framework per la descrizione ad alto livello dei problemi il cui scopo si sovrapponeva a quello del'UFL), FErari (usato per la generazione di spazi di funzione) e Dorsal (usato per automatizzare l'installazione di FEniCS).

Moduli esterni

FEniCS offre molte interfacce per interagire con i pacchetti di algebra lineare, che nella nomenclatura di FEniCS sono detti *linear algebra backends*. L'implementazione usata per il precondizionamento e la risoluzione delle matrici lineari è a discrezione dell'utente. FEniCS cerca quella più consona fra quelle disponibili sulla piattaforma. La scelta di default è PETSc[58], dato che la maggior parte delle piattaforme (in particolare in ambiente Unix) la supporta. In alternativa sono impiegati uBLAS[59], Trilinos/Epetra[60] e MTL4[61]. L'utente può comunque impiegare un backend custom, anche basato su elaborazione parallela su GPU o distribuita[64].

Al momento, tutti e quattro i backend citati offrono funzionalità per il precondizionamento e la risoluzione sia diretta sia iterativa di sistemi di matrici dense, ma solo uBLAS offre delle API specifiche per il trattamento di matrici e vettori sparsi. Dato che le funzionalità sono uniformi e l'interfaccia offerta è pressoché identica per tutti i backends

è molto semplice per l'utente scegliere. Per scegliere un backend in Python è sufficiente scrivere

```
dolfin.parameters['linear_algebra_backend'] = backend
```

che ha come corrispettivo in C++

```
dolfin::parameters['linear_algebra_backend'] = backend;
```

dove *backend* è una stringa che rappresenta il backend scelto come 'PETSc', 'uBLAS', 'Epetra' o anche 'MTL4'.

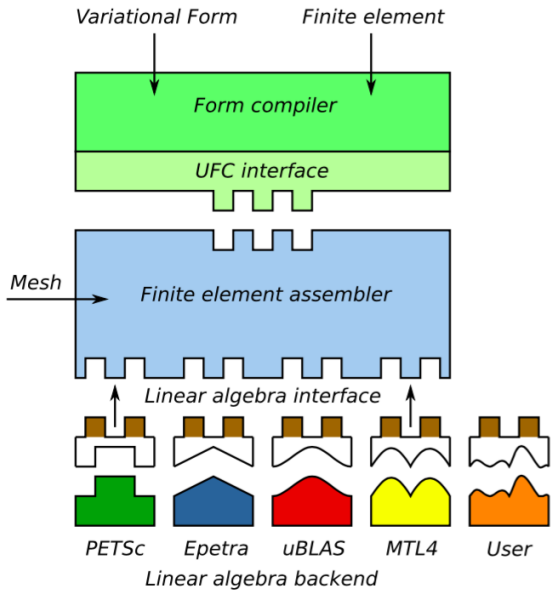


Figura 3.6: FEniCS può integrare molteplici liberie di calcolo. Tratto da [62]

3.4.2 Applicazioni

L'uso della FEM è ampiamente diffuso in applicazioni commerciali per la facilità con cui questo tipo di analisi può essere adattato a strutture complesse e irregolari senza la difficoltà che altre tecniche hanno nell'imposizione delle condizioni al contorno. L'aspetto più importante del successo di FEniCS rispetto ad altre librerie FEM open source gratuite è nell'ottimo compromesso fra il tempo impiegato dall'utente per descrivere un problema e quello necessario alla macchina per risolverlo, grazie all'uso dell'UFL e della generazione automatica di codice.

Grazie alla sua flessibilità, FEniCS è stato impiegato per la valutazione della risposta meccanica di materiali compositi[72], la cui simulazione richiede l'impiego di complessi modelli non standard. Su FEniCS

è basato anche un framework chiamato Magnum.fe[73][74] creato appositamente per descrivere fenomeni di micromagnetismo all'interno di materiali ferromagnetici.

Esiste un ampio spettro di applicazioni in campo ingegneristico dallo studio della propagazione di campi elettromagnetici generati dalle antenne[75] fino alla risoluzione di classici problemi di fluido dinamica computazionale (CDF) come il moto turbolento attorno alle ali di un velivolo[76]. In questa classe di problemi il punto di forza di FEniCS è la facilità con cui si possono importare strutture disegnate con CAD.

In ambito scientifico, FEniCS è stato impiegato anche per la simulazione del moto del magma nel mantello terrestre[77], in cui è importante computare sia l'effetto dei flussi di calore e della gravità sia la densità e visosità che dipendono dalle condizioni di temperatura e pressione (e che in certe condizioni lo rendono un liquido non newtoniano).

Una delle prime applicazioni di FEniCS è stata la simulazione del movimento meccanico delle valvole cardiache durante il battito[78], ma in campo medico è stato impiegato anche per progetti più avanzati come lo studio dell'aneurisma aortico[79]. Un altro progetto all'avanguardia che impiega FEniCS è lo studio di nuove terapie antitumorali basate sul riscaldamento locale indotto mediante campi magnetici a bassa frequenza[80] (ipertermia magnetica) che richiedono una accurata descrizione delle complesse interazioni fra il campo elettromagnetico e il tessuto.

Va notato infine che le componenti di FEniCS sono sotto licenza GNU LGPL v3, che permette l'utilizzo di questa libreria sia in altri progetti open source sia all'interno di software proprietario[81].

Capitolo 4

Modelli matematici e fisici

Per simulare il comportamento del dispositivo è stato necessario descrivere il comportamento dei portatori sotto l'effetto del campo elettrico e delle radiazioni ionizzanti. Per trovare la soluzione approssimata esiste un ampio spettro di tecniche usate, come le Differenze Finite (FDM) o i Volumi Finiti (FVM), ma per la realizzazione di questo progetto si è scelto di utilizzare il metodo degli Elementi Finiti (FEM). In questo capitolo verrà introdotto il metodo degli elementi finiti e giustificata questa scelta. Successivamente saranno mostrate le relazioni costitutive del dispositivo impiegate per la simulazione dei fenomeni di ionizzazione all'interno dei circuiti integrati.

4.1 Metodo degli Elementi Finiti

4.1.1 Introduzione e definizione

Per semplicità consideriamo il problema classico di Poisson, dove è necessario cercare una funzione u tale per cui $\nabla^2 \mathrm{u} = -f$. Definiamo un dominio Ω su cui è definita la funzione scalare f e la cui frontiera è $\partial\Omega$.

$$\begin{aligned} \nabla^2 \mathrm{u} &= -f \quad && \text{in} \quad \Omega \\ \mathrm{u} &= \mathrm{u}_0 \quad && \text{in} \quad \partial\Omega_D \subset \partial\Omega \\ \partial \mathrm{u}/\partial n &= -g \quad && \text{in} \quad \partial\Omega_N \subset \partial\Omega \end{aligned}$$

Su $\partial\Omega_D$ è definita una condizione di Dirichlet ed impone il valore della funzione u sul confine del dominio. Su $\partial\Omega_N$ la condizione di Von Neumann impone la derivata parziale lungo la direzione normale alla frontiera. Ignoriamo altre tipologie di condizioni al contorno più generali per non appesantire la trattazione. Affinché il problema abbia una

soluzione univoca ipotizziamo che $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ e che ci sia almeno un punto in cui il valore della funzione è noto (e quindi che $\partial\Omega_D \neq \{\}$)[1]. Questa forma del problema è detta classica o *formulazione forte*.

Moltiplichiamo la relazione $\nabla^2 \mathrm{u} = -f$ per la funzione di test v e integriamo su tutto il dominio Ω:

$$\int_\Omega \nabla^2 \mathrm{u}\, \mathrm{v}\, \mathrm{d}\omega = -\int_\Omega f \mathrm{v}\, \mathrm{d}\omega$$

Si applica la prima identità di Green[2]:

$$\int_\Omega \nabla \mathrm{u} \cdot \nabla \mathrm{v}\, \mathrm{d}\omega = \int_\Omega f \mathrm{v}\, \mathrm{d}\omega + \int_{\partial\Omega} \frac{\partial \mathrm{u}}{\partial n} \mathrm{v}\, \mathrm{d}s$$

Notare che dal punto di vista matematico con questa formulazione non è richiesto che u e v siano differenziabili. Una condizione sufficiente è che siano integrabili nel dominio richiesto, ovvero che siano entrambi di classe $H^1(\Omega)$.

Definiamo lo spazio di funzioni di prova V, sfruttando il fatto che dove è imposta una condizione di Dirichlet (ovvero lungo $\partial\Omega_D$) il valore della soluzione u è nota:

$$V = \{\mathrm{u} \in H^1(\Omega) : \mathrm{u} = \mathrm{u}_0 | \partial\Omega_D\}$$

Imponiamo per ipotesi che la funzione di test sia nulla lungo $\partial\Omega_D$. Definiamo quindi lo spazio di funzioni di test $\hat{V}$:

$$\hat{V} = \{\mathrm{v} \in H^1(\Omega) : \mathrm{v} = 0 | \partial\Omega_D\}$$

Per come è stato costruito $\hat{V}$ e dato che $\partial\Omega - \partial\Omega_D = \partial\Omega_N$ si ha:

$$\int_{\partial\Omega} \frac{\partial \mathrm{u}}{\partial n} \mathrm{v}\, \mathrm{d}s = \int_{\partial\Omega_N} \frac{\partial \mathrm{u}}{\partial n} \mathrm{v}\, \mathrm{d}s = -\int_{\partial\Omega_N} g\, \mathrm{v}\, \mathrm{d}s$$

[1] È evidente che esistono condizioni diverse da quelle di Drichlet che sono sufficienti per definire una soluzione unica (un esempio banale è $\int_\Omega \mathrm{u}\, \mathrm{d}\omega = 0$ con $\partial\Omega = \partial\Omega_N$). Tuttavia, se vi sono solo condizioni sulla derivata al contorno la soluzione è definibile soltano a meno di una costante addittiva.

[2] La prima identità di Green afferma che:

$$\int_{\partial\Omega} \phi \frac{\partial \psi}{\partial n}\, \mathrm{d}s = \int_\Omega \nabla\phi \cdot \nabla\psi\, \mathrm{d}\omega + \int_\Omega \phi\, \nabla^2\psi\, \mathrm{d}\omega$$

La regola di Green può essere considerata una estensione a più dimensioni della regola di integrazione per parti. Se ipotizziamo di avere un dominio monodimensionale $[a, b]$ e poniamo $g = \frac{\mathrm{d}G}{\mathrm{d}x} = \phi$ e $f = \frac{\mathrm{d}F}{\mathrm{d}x} = \nabla^2\psi$, si ha:

$$[GF]_a^b = \int_a^b \frac{\mathrm{d}(GF)}{\mathrm{d}x}\, \mathrm{d}x = \int_a^b gF\, \mathrm{d}x + \int_a^b G\, f\, \mathrm{d}x$$

Sotto queste ipotesi, si ricava:

$$\int_\Omega \nabla \mathrm{u} \cdot \nabla \mathrm{v} \, \mathrm{d}\omega = \int_\Omega f \mathrm{v} \, \mathrm{d}\omega - \int_{\partial\Omega_N} g \, \mathrm{v} \, \mathrm{d}s$$

Questa forma è convenzionalmente detta *formulazione debole*, in contrapposizione a quella forte in cui il problema è stato inizialmente posto.

Notare che le condizioni di Dirichlet influenzano solo la definizione dello spazio delle funzioni di prova, a differenza delle condizioni di Neumann che rientrano nell'espressione del problema variazionale.

Per risolvere numericamente l'equazione differenziale, è necessario limitare gli insiemi delle funzioni di prova e di test a due sotto spazi di dimensione finita N detti rispettivamente V_h e $\hat{V}_h$.

La famiglia della base dello spazio vettoriale è caratterizzata dal loro andamento spaziale. In generale, ogni funzione generatrice è caratterizzata da un sottodominio di riferimento. Nel caso di sottodomini tridimensionali si usa il termine di *voxel* per indicare la cella elementare, in contrapposizione al termine *pixel*, che è il corrispettivo nel caso bidimensionale. L'organizzazione dei sottodomini è rappresentata dal mesh. La famiglia inoltre può essere caratterizzata da un ordine, per distinguerla da altre famiglie di funzioni simili. Il mesh e la famiglia di funzioni definiscono univocamente la discretizzazione del sistema.

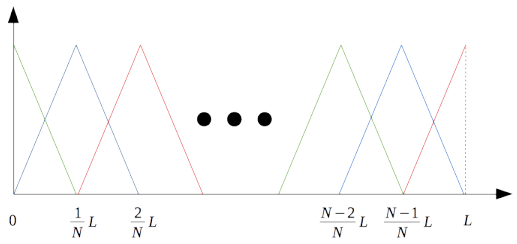


Figura 4.1: Esempio di famiglia di funzioni lineari a tratti del primo ordine su un mesh monodimensionale regolare. Notare che questa discretizzazione è equivalente a quella data dal metodo delle differenze finite al primo ordine.

Il problema da risolvere diventa trovare la funzione u_h (detta *soluzione debole*) tale per cui:

$$\mathrm{u}_h \in V_h : \quad \int_\Omega \nabla \mathrm{u}_h \cdot \nabla \mathrm{v}_h \, \mathrm{d}\omega = \int_\Omega f \mathrm{v}_h \, \mathrm{d}\omega - \int_{\partial\Omega_N} g \, \mathrm{v}_h \, \mathrm{d}s \quad \forall \mathrm{v}_h \in \hat{V}_h$$

Il membro a sinistra è bilineare rispetto a v_h e u_h, mentre il termine a destra è lineare solo rispetto a v_h.

La scelta dello spazio di funzioni di test $\hat{V}_h$ è determinato dal particolare problema da risolvere. Una possibilità è impiegare delle funzioni campione di dirac nei nodi del mesh. Questa tecnica porta a minimizzare l'errore solo ai nodi, ma può essere conveniente in sistemi con forte periodicità e per usare metodi spettrali o pseudo spettrali. Per minimizzare il modulo assoluto dell'errore su un intero sottodominio si può ricorrere ad una base di funzioni di test costanti a tratti.

Tuttavia, spesso in ambito ingegneristico è importante minimizzare l'energia associata all'errore piuttosto che il suo modulo. A questo scopo si impiega la medesima base di funzioni per generare sia V_h sia $\hat{V}_h$ all'interno del dominio. Questa tecnica è detta *Metodo di Galerkin.*

Poniamo come basi di $\hat{V}_h$ e V_h rispettivamente le funzioni $\phi_1, ..., \phi_N$ e $\hat{\phi}_1, ..., \hat{\phi}_N$. Di conseguenza:

$$\mathrm{u}_h = \sum_{i=1}^{N} u_i \phi_i$$

$$\mathrm{v}_h = \sum_{j=1}^{N} v_j \hat{\phi}_j$$

Il problema discretizzato diventa:

$$\int_\Omega \nabla \sum_{i=1}^{N} u_i \phi_i \cdot \nabla \sum_{j=1}^{N} v_j \hat{\phi}_j \,\mathrm{d}\omega = \int_\Omega f \sum_{j=1}^{N} v_j \hat{\phi}_j \,\mathrm{d}\omega - \int_{\partial\Omega_N} g \sum_{j=1}^{N} v_j \hat{\phi}_j \,\mathrm{d}s$$

Ovvero:

$$\sum_{i=1}^{N} u_i \int_\Omega \nabla\phi_i \cdot \nabla\hat{\phi}_j \,\mathrm{d}\omega = \int_\Omega f\hat{\phi}_j \,\mathrm{d}\omega - \int_{\partial\Omega_N} g\,\hat{\phi}_j \,\mathrm{d}s \quad \forall j \in \{1, ..., N\}$$

Di conseguenza, trovare la proiezione u_h sullo spazio $\hat{V}_h$ della soluzione u equivale a risolvere il sistema lineare

$$Au = b$$

dove

$$A_{ij} = \int_\Omega \nabla\phi_i \cdot \nabla\hat{\phi}_j \,\mathrm{d}\omega$$

$$b_i = \int_\Omega f\hat{\phi}_i \,\mathrm{d}\omega - \int_{\partial\Omega_N} g\,\hat{\phi}_i \,\mathrm{d}s$$

Il vettore b è spesso chiamato *vettore di carico* (Load vector) e la matrice A è detta *matrice di rigidezza del sistema* (Stiffness matrix) per il loro significato in meccanica delle strutture. È importante notare che

per come sono state scelte le basi di $\hat{V}_h$ e V_h la matrice è hermitiana e definita positiva indipendentemente dalle caratteristiche del problema specifico. Questa peculiarità permette di applicare delle tecniche di precondizionamento come la decomposizione di Cholesky[3].

Si può dimostrare che sotto queste ipotesi, u_h è la proiezione ortogonale della soluzione esatta u sullo spazio V_h.

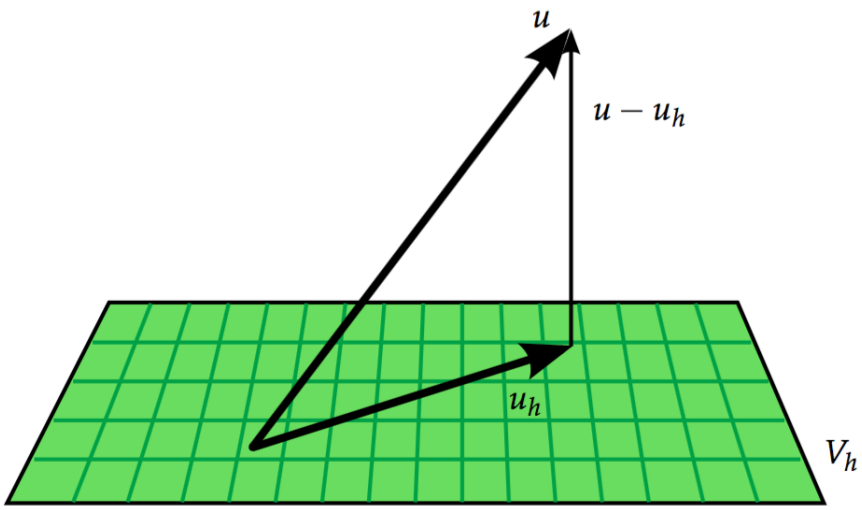


Figura 4.2: Nel metodo di Galerkin, la soluzione approssimata u_h è la proiezione ortogonale della soluzione u sullo spazio V_h. Tratto da [62]

L'algoritmo risultante è facile da adattare ad una ampia classe di problemi e gode della proprietà di minizzare l'energia dell'errore sull'intero dominio.

4.1.2 Analisi della stabilità e dell'errore

Esistenza ed unicità

Per dimostrare che la soluzione è unica si può ricorrere al teorema di Lions-Lax-Milgram[82].

Imponiamo che gli spazi V e $\hat{V}$ siano spazi di Hilbert e che, come richiesto dal metodo di Galerkin, $V = \hat{V}$ all'interno del dominio Ω. La dimostrazione può essere estesa anche a metodi che non rispettano quest'ultima condizione[83].

[3] Un appropriato precondizionamento permette di accelerare notevolemente la ricerca della soluzione. La scelta dipende dalle caratteristiche del problema e dalla piattaforma impiegata. Le librerie di calcolo più diffuse come PETSc[58] o uBLAS[59] supportano una ampia varietà di tecniche di prendicondizionamento.

Definiamo una generica forma bilineare a tale che

$$a : V \times V \to \mathbb{R}$$

Definiamo una funzione lineare L tale che

$$L : V \to \mathbb{R}$$

Imponiamo che siano continue, ovvero che esista una costante reale $\gamma > 0$ tale per cui

$$a(\mathrm{u}, \mathrm{v}) \leq \gamma \|\mathrm{u}\|_V \|\mathrm{v}\|_V \quad \forall \mathrm{u}, \mathrm{v} \in V$$

$$L(\mathrm{v}) \leq \gamma \|\mathrm{v}\|_V \quad \forall \mathrm{v} \in V_h$$

Imponiamo che a sia una forma *coerciva*, ovvero che esista una costante reale $\alpha > 0$ tale per cui

$$a(\mathrm{u}, \mathrm{u}) \geq \alpha \|\mathrm{u}\|_V^2 \quad \forall \mathrm{u}, \mathrm{v} \in V$$

Per come è definito il sistema considerato, è evidente che:

$$a(\mathrm{u}, \mathrm{v}) = \int_\Omega \nabla \mathrm{u} \cdot \nabla \mathrm{v} \, \mathrm{d}\omega$$

$$L(\mathrm{v}) = \int_\Omega f \mathrm{v} \, \mathrm{d}\omega - \int_{\partial\Omega_N} g \, \mathrm{v} \, \mathrm{d}s$$

È facile dimostrare che L e a sono forme continue. Per la coercività si può notare che ad a corrisponde una matrice *simmetrica definita positiva*: la funzione $a(\mathrm{v}, \mathrm{v})$ a possiede solo autovalori positivi e quindi si annulla se e solo se $\|\mathrm{v}\|_V = 0$. Di conseguenza, basta porre α pari al più piccolo autovalore di $a(\mathrm{v}, \mathrm{v})$ affinché la diseguaglianza richiesta sia rispettata.

Sotto queste condizioni, il teorema di Lions-Lax-Milgram permette di affermare che esiste una sola soluzione al problema in forma debole[82][63].

Stima dell'errore

Una volta dimostrato che il metodo agli elementi finiti fornisce una soluzione approssimata valida e univoca, il passo successivo è valutare l'errore commesso.

Una stima può essere ottenuta a priori, senza esplicitamente calcolare u_h, sfruttando il *lemma di Cea*. Per la coercività su a si ha:

$$\alpha \|\mathrm{u} - \mathrm{u}_h\|_V^2 \leq a(\mathrm{u} - \mathrm{u}_h, \mathrm{u} - \mathrm{u}_h) = a(\mathrm{u} - \mathrm{u}_h, \mathrm{u} - \mathrm{v}) + a(\mathrm{u} - \mathrm{u}_h, \mathrm{v} - \mathrm{u}_h)$$

Dato che $(\mathrm{v} - \mathrm{u}_h) \in \hat{V}_h$ e che

$$a(\mathrm{u} - \mathrm{u}_h, \mathrm{v}) = a(\mathrm{u}, \mathrm{v}) - a(\mathrm{u}_h, \mathrm{v}) = L(\mathrm{v}) - L(\mathrm{v}) = 0 \quad \forall \mathrm{v} \in \hat{V}_h$$

si ottiene

$$\alpha \|\mathrm{u} - \mathrm{u}_h\|_V^2 \leq a(\mathrm{u} - \mathrm{u}_h, \mathrm{u} - \mathrm{v}) \leq \gamma \|\mathrm{u} - \mathrm{v}\|_V \quad \forall \mathrm{v} \in \hat{V}_h$$

e si ricava infine il lemma di Cea:

$$\|\mathrm{u} - \mathrm{u}_h\|_V \leq \frac{\gamma}{\alpha} \|\mathrm{u} - \mathrm{v}\|_V \quad \forall \mathrm{v} \in \hat{V}_h \tag{4.1}$$

Se si definisce il modulo nello spazio $\hat{V}$ come $\|\mathrm{u}\|_{\hat{V}} = \sqrt{a(\mathrm{u}, \mathrm{u})}$ e si pone $\alpha = \gamma = 1$ si ottiene:

$$\|\mathrm{u} - \mathrm{u}_h\|_V \leq \|\mathrm{u} - \mathrm{v}\|_V \quad \forall \mathrm{v} \in \hat{V}_h \tag{4.2}$$

Di conseguenza, u_h è la proiezione secondo a della soluzione u sullo spazio $\hat{V}_h$.

Un'altra possibilità è valutare l'errore è a posteriori, una volta calcolata la soluzione approssimata u_h.

Definiamo il residuo duale come

$$r(\mathrm{v}) = L(\mathrm{v}) - a(\mathrm{u}_h, \mathrm{v})$$

Per la continuità e bilinearità di a si ha:

$$r(\mathrm{v}) = a(\mathrm{u}, \mathrm{v}) - a(\mathrm{u}_h, \mathrm{v}) = a(\mathrm{u} - \mathrm{u}_h, \mathrm{v}) \leq \gamma \|\mathrm{u} - \mathrm{u}_h\|_V \|\mathrm{v}\|_V$$

Per la coercività di a si ha

$$\alpha \|\mathrm{u} - \mathrm{u}_h\|_V^2 \leq a(\mathrm{u} - \mathrm{u}_h, \mathrm{u} - \mathrm{u}_h) = L(\mathrm{u} - \mathrm{u}_h) - a(\mathrm{u}_h, \mathrm{u} - \mathrm{u}_h) = r(\mathrm{u} - \mathrm{u}_h)$$

Unendo le due disequazioni si ricava una stima a posteriori dell'errore commesso

$$\alpha \|\mathrm{u} - \mathrm{u}_h\|_V \leq \frac{1}{\|\mathrm{v}\|_V} \|r(\mathrm{v})\|_V \leq \gamma \|\mathrm{u} - \mathrm{u}_h\|_V \tag{4.3}$$

Se si pone $\alpha = \gamma = 1$, il massimo di $\|r(\mathrm{v})\|_V$ è pari alla *energia massima dell'errore.*

Adattamento automatico

La stima dell'errore serve per valutare l'accuratezza di un risultato fornito da un algoritmo. Se non è sufficiente allo scopo, si può ripetere la procedura con un altro algoritmo che garantisca un errore minore fino a raggiungere la tolleranza desiderata.

La tecnica degli elementi finiti permette di automatizzare in parte o del tutto questo procedimento. Per aumentare l'accuratezza è sufficiente scegliere un set di funzioni generatrici tale per cui il residuo debole abbia modulo inferiore a quello richiesto. Esistono tre strategie principali per la costruzione di una discretizzazione del dominio che permetta di ridurre l'errore.

p-FEM: incrementando l'ordine polinomiale delle funzioni l'errore la soluzione può essere approssimata più fedelmente.

h-FEM: aumentando la densità di celle del mesh nelle zone in cui l'errore totale cala.

hp-FEM: combinando i due approcci precedenti si ottiene una tecnica più flessibile, anche se più complessa da implementare.

L'ultima strategia consente una ricerca della base adatta più veloce delle altre due[84].

In altre parole, nella cosidetta hp-FEM si parte da una stima dell'errore della soluzione ottenuta con pochi punti nel mesh e un ordine delle funzioni basso. Se l'errore commesso supera la tolleranza richiesta, si aumenta la definizione nelle zone del mesh più soggette ad errore oppure si varia l'ordine delle funzioni adottate. Si tratta di un compromesso: ad una maggiore risoluzione e ad un maggior ordine corrisponde una maggiore accuratezza, ma anche un incremento delle risorse hardware impiegate. La piattaforma software adottata in questo progetto consente di definire un criterio per il raffinamento del mesh (h-FEM) in maniera relativamente semplice[63], come sarà mostrato nell'ultimo capitolo.

4.1.3 Confronto con altri metodi

Come è stato fatto notare, la formulazione debole pone delle condizioni meno restrittive sulla soluzione rispetto alla formulazione forte. Possono essere discontinue, consentire approssimazioni esponenziali o essere definite ad hoc per una classe opportuna di problemi.

L'aspetto più interessante è anche altri tipi di metodi possono essere rappresentati come casi particolare di FEM. Impiegare un algoritmo

generale come la FEM può portare ad un overhead rispetto ad una implementazione specifica, che potrebbe sfruttare ottimizzazioni a basso livello come il caching degli operandi oppure evitare la costruzione e la memorizzazione delle matrici necessarie per la definizione delle basi degli spazi di prova e test. Nonostante ciò, il beneficio di avere un metodo generale facile da adattare a nuove situazioni e meno prono ad errori in fase di programmazione è sufficiente a giustificare il successo della analisi agli elementi finiti in campo ingegneristico.

Metodo delle Differenze Finite

Il metodo delle differenze finite si basa sulla approssimazione della serie di Taylor della soluzione forte:

$$f(x+h) = f(x) + \frac{f^1(x)}{1!}h + \frac{f^2(x)}{2!}h^2 + ... + \frac{f^n(x)}{n!}h^n + O(h^n)$$

Sovrapponendo i contributi dei nodi vicini di una griglia regolare si ricava un sistema di equazioni. Per esempio, sotto queste ipotesi l'equazione di Poisson al primo ordine diventa:

$$\frac{\psi_{k-1} + \psi_{k+1} - 2\psi_k}{h^2} = -\frac{\rho_k}{\varepsilon}$$

Aumentando l'ordine a cui si approssima la funzione e aumentando la risoluzione della griglia si riduce l'errore. È un metodo semplice e robusto, ma poco flessibile: l'adattamento a mesh irregolari è più complesso rispetto a quanto possibile con il metodo degli elementi finiti. Oltre ad essere il metodo più antico e più facile da analizzare e implementare, le differenze finite sono lo standard nella simulazione di transistori nel tempo.

Metodo dei Volumi Finiti

In molti modelli fisici è possibile identificare una grandezza che rispetta una *legge di conservazione*. Un esempio può essere la concentrazione dei portatori nel modello drift-diffusion:

$$\sum_k J_{ik} = -\frac{\partial p_i}{\partial t}\Delta V_i + (G_i - R_i)\Delta V_i$$

Il sistema viene risolto rispetto ai flussi fra le celle. Oltre a rendere naturale l'imposizione delle condizioni al contorno sul flusso, il grande vantaggio della proprietà di conservazione è che gli errori di discretizzazione si compensano fra le celle celle adiacenti: in altre parole, se

vi è un errore ΔJ_{ij} nel calcolo del flusso J_{ij} fra la cella i-esima e la cella j-esima, l'errore sulla quantità di interesse è compensate fra le due celle e quindi l'integrale della quantità sull'intero dominio rimane corretto. Per questo motivo il metodo dei volumi finiti è popolare in fluidodinamica computazionale e viene impiegato in alcuni moduli misti FEM-FVM di simulazione di semiconduttori, per esempio nell'ambiente Sentaurus di Synopsys[53]. Il fatto di basarsi sul calcolo dei flussi rende questo metodo poco preciso in presenza di derivate di ordine elevato e in generale poco adatto in presenza di forti discontinuità.

Esistono famiglie di elementi finiti che consentono di rispettare la legge di conservazione, permettendo quindi di implementare un sistema di equazioni ai volumi finiti all'interno di una simulazione FEM[85].

Metodi spettrali

Al contrario del FVM, i metodi spettrali sono particolarmente efficaci nella risoluzione di equazioni differenziali di ordine elevato. Anche i metodi spettrali sono implementabili con famiglie di elementi finiti in cui sia rispettata la condizione di *ortogonalità* degli elementi della base:

$$\int_\Omega \phi_i \phi_j \, \mathrm{d}\omega \begin{cases} = 0 & i \neq j \\ \neq 0 & i = j \end{cases} \tag{4.4}$$

Il metodo spettrale più conosciuto è la serie di Fourier, ma anche i polinomi di Chebyshev o di Legendre possono essere sfruttate come basi. Dato che queste funzioni prese singolarmente sono non nulle in ampie porzioni del dominio, i metodi spettrali sono detti a volte *approssimazioni globali* per contrapporli alle *approssimazioni locali* degli altri metodi. Una conseguenza è che in assenza di forti irregolarità l'accuratezza può essere molto elevata anche con basi piccole. Gli svantaggi che determinano lo scarso utilizzo in ambito ingegneristico sono la poca adattabilità in mesh irregolari e il costo computazionale relativamente alto.

4.2 Fisica del dispositivo

All'interno di un materiale generico l'andamento del potenziale elettrostatico ψ segue le equazioni di Maxwell[4] ed è descritto dalla legge di Poisson

$$\nabla \cdot (\varepsilon \nabla \psi) = -\rho = -q \sum_{j=1}^{n} z_j C_j$$

[4] $\nabla \cdot \overrightarrow{D} = \rho$ e $\overrightarrow{D} = \varepsilon \overrightarrow{E}$, dove per definizione $\overrightarrow{E} = -\nabla \psi$

dove z_j e C_j indicano rispettivamente la valenza e la concentrazione della specie j-esima. Nel problema considerato, la costante dielettrica del silicio è uniforme e le uniche specie chimiche presenti oltre agli elettroni e alle lacune sono i droganti N_a e N_d.

$$\nabla^2\psi = -\frac{\rho}{\varepsilon} = -\frac{q}{\varepsilon}(N_d - N_a + p - n)$$

Nel caso generale in cui la mobilità dei portatori sia uniforme e la velocità media sia bassa, dalla prima legge di Fick e dalla legge di Ohm si ottengono le densità di flusso di diffusione e di drift dei portatori da cui è banale ricavare la densità di corrente dovuta agli elettroni e alle lacune

$$\overrightarrow{J}_n = -q(\overrightarrow{F}_{diff_n} + \overrightarrow{F}_{drift_n}) = qD_n\nabla n - q\mu_n n\nabla\psi \tag{4.5}$$

$$\overrightarrow{J}_p = q(\overrightarrow{F}_{diff_p} + \overrightarrow{F}_{drift_p}) = -qD_p\nabla p - q\mu_p p\nabla\psi \tag{4.6}$$

Infine, per completare la descrizione si può impiegare la legge di continuità[5]:

$$\frac{\partial n}{\partial t} = \frac{1}{q}\nabla\cdot\overrightarrow{J}_n + G - R \tag{4.7}$$

$$\frac{\partial p}{\partial t} = -\frac{1}{q}\nabla\cdot\overrightarrow{J}_p + G - R \tag{4.8}$$

dove G e R indicano rispettivamente la frequenza di generazione e ricombinazione delle coppie elettrone lacuna. Considerando solo effetti termici e trascurando quelli al secondo ordine (come ricombinazione Auger) o legati alla presenza di stati trappola intermedi (Shockley–Read–Hall) si ottiene:

$$R = \frac{pn}{\tau}$$

$$G = \frac{n_i^2}{\tau}$$

dove τ è il tempo di vita medio dei portatori mentre n_i è la concentrazione di portatori all'equilibrio nel semiconduttore intrinseco. Per campi elettrici di ridotta intensità la generazione non dipende dalle altre grandezze: se il campo elettrico è vicino al valore di break down l'effetto di moltiplicazione può essere schematizzato in prima approssimazione come una generazione dipendente dal campo elettrico e dalla concentrazione di portatori, ma questa situazione non verrà presa in

[5] $\frac{\partial \mathrm{u}}{\partial t} = -\nabla\cdot\overrightarrow{F} + G - R$

considerazione in questa tesi e le uniche forme di generazione considerate sono di origine termica o indotte da radiazioni ionizzanti. In assenza di radiazioni ionizzanti la generazione di origine termica è costante e pari a $\frac{n_i^2}{\tau}$.

Sostituendo le espressioni (4.5) e (4.6) in (4.7) e (4.8) si ricava:

$$\frac{\partial n}{\partial t} = (\nabla D_n \cdot \nabla n + D_n \nabla^2 n) - (n \nabla \mu_n \cdot \nabla \psi + \mu_n \nabla n \cdot \nabla \psi + \mu_n n \nabla^2 \psi) + (G - R)$$

$$\frac{\partial p}{\partial t} = (\nabla D_p \cdot \nabla p + D_p \nabla^2 p) + (p \nabla \mu_p \cdot \nabla \psi + \mu_p \nabla p \cdot \nabla \psi + \mu_p p \nabla^2 \psi) + (G - R)$$

Per ipotesi i coefficienti di mobilità e diffusione sono uniformi. Ponendo $V_t = \frac{kT}{q}$ e sfruttando la relazione di Einstein $D = \mu V_t$ si ricava:

$$\nabla^2 \psi = -\frac{q}{\varepsilon}(N_d - N_a + p - n) \tag{4.9}$$

$$\frac{\partial n}{\partial t} = \mu_n (V_t \nabla^2 n - \nabla n \cdot \nabla \psi - n \nabla^2 \psi) + (G - R) \tag{4.10}$$

$$\frac{\partial p}{\partial t} = \mu_p (V_t \nabla^2 p + \nabla p \cdot \nabla \psi + p \nabla^2 \psi) + (G - R) \tag{4.11}$$

$$G - R = \frac{n_i^2 - pn}{\tau} \tag{4.12}$$

4.3 Transitorio nel tempo

Se il tempo di campionamento usato è sufficientemente piccolo anche la variazione del profilo della carica è piccola e quindi anche la variazione del campo elettrico fra un istante e il successivo è piccola. È legittimo quindi calcolare la distribuzione dei portatori utilizzando l'andamento del potenziale e della funzione generazione-ricombinazione calcolata al passo precedente.

$$p_k, n_k \to \psi_k$$

$$p_k, n_k \to R_k$$

$$n_k, \psi_k, R_k \to n_{k+1}$$

$$p_k, \psi_k, R_k \to p_{k+1}$$

Utilizzando questa approssimazione le equazioni sono disaccoppiate e lineari.

Per discretizzare l'andamento dei portatori è necessario ricorrere al metodo alle differenze finite. Una vasta famiglia di metodi è quella di Runge–Kutta, che comprende discretizzazioni sia implicite sia espicite

di vario ordine. Senza entrare nel dettaglio, il metodo classico di Runge-Kutta è un metodo alle differenze finite del quarto ordine esplicito. Utilizzare approssimazioni di ordine superiore riduce l'errore commesso a parità di passo di discretizzazione, a costo di un maggiore costo computazionale. Come compromesso è stata scelta la discretizzazione di Crank–Nicolson, un metodo implicito che può essere considerato come la media del risultato dell'applicazione del metodo di Eulero esplicito e implicito.
Posto $f = \frac{\partial \mathrm{u}}{\partial t}$ si ha:

$$u_{k+1} = u_k + f(u_k)\Delta t \qquad \text{Eulero esplicito in avanti}$$

$$u_{k+1} = u_k + f(u_{k+1})\Delta t \qquad \text{Eulero implicito all'indietro}$$

$$u_{k+1} = u_k + \frac{f(u_k) + f(u_{k+1})}{2}\Delta t \qquad \text{Crank–Nicolson}$$

Rispetto agli altri due metodi, la discretizzazione secondo il metodo di Crank–Nicolson offre sia una buona accuratezza (l'errore di approssimazione è $O(\Delta t^2)$) sia stabilità incondizionata[6]. Definiamo la media fra il profilo dei portatori fra l'istante i e $i+1$:

$$n_{i+\frac{1}{2}} = \frac{n_i + n_{i+1}}{2}$$
$$p_{i+\frac{1}{2}} = \frac{p_i + p_{i+1}}{2}$$

Otteniamo le equzioni fondamentali discretizzate con Crank-Nicolson:

$$\nabla^2\psi_k = -\frac{q}{\varepsilon}(N_d - N_a + p_k - n_k) \tag{4.13}$$

$$R_k = \frac{p_k n_k}{\tau} \tag{4.14}$$

$$\frac{n_{k+1}-n_k}{\Delta t} = \mu_n(V_t\nabla^2 n_{k+\frac{1}{2}} - \nabla n_{k+\frac{1}{2}}\cdot\nabla\psi_k - n_{k+\frac{1}{2}}\nabla^2\psi_k) + G - R_k \tag{4.15}$$

$$\frac{p_{k+1}-p_k}{\Delta t} = \mu_p(V_t\nabla^2 p_{k+\frac{1}{2}} + \nabla p_{k+\frac{1}{2}}\cdot\nabla\psi_k + p_{k+\frac{1}{2}}\nabla^2\psi_k) + G - R_k \tag{4.16}$$

Se ad un certo passo k si verifica un evento come il passaggio di una radiazione ionizzante, ai portatori deve essere aggiunto un termine legato alla generazione, descritto per esempio dalla formula di Bethe presentata in precedenza nella sottosezione 2.1.2.

[6]Rispetto a Crank–Nicolson, il metodo di Eulero all'indietro offre *L-stabilità*, che grantisce l'assenza di oscillazione spurie alle alte frequenze. Il metodo di Eulero in avanti soffre sia di scarsa stabilità sia di convergenza lenta, l'unico vantaggio è la velocità di esecuzione

Dato che il sistema è stato disaccoppiato in equazioni che prese singolarmente sono lineari, il problema è risolvibile senza particolari difficoltà, a patto che il passo di discretizzazione temporale Δt e quello di discretizzazione spaziale siano sufficientemente piccoli rispetto alla costante di diffusione e alla velocità di drift dei portatori. Per valutare questi parametri esiste un ampio spettro di condizioni che dipendono da come la discretizzazione è stata effettuata. Nelle equazioni appena presentate la discretizzazione non è ancora esplicitata perché dipendende dal mesh e dalla famiglia di elementi.

In aggiunta, va notato che le matrici associate al problema cambiano ad ogni iterazione, ma in certe condizioni (come nel caso in cui il contributo al potenziale elettrostatico dato dalle cariche libere sia trascurabile) è possibile scomporre le matrici relative al calcolo dei portatori in due matrici triangolari, di cui una costante e l'altra da aggiornare ad ogni passo. Dato che per la descrizione nel tempo del sistema sono necessari molte iterazioni, questa può essere una ottimizzazione significativa.

4.4 Simulazione a regime

Per simulare il comportamento del dispositivo a regime il metodo più intuitivo è usare le relazioni già identificate per il problema in transitorio e simulare l'evoluzione del sistema su un tempo sufficentemente lungo. Questa tecnica è robusta ma estremamente dispendiosa, quindi in generale conviene risolvere il sistema non lineare. Il sistema può essere risolto usando come variabili il potenziale elettrico e le concentrazioni dei portatori come è stato fatto per la descrizione in transitorio, ma si possono scegliere altre combinazioni di variabili indipendenti[7].

In presenza di un disturbo (per esempio l'applicazione di una differenza di potenziale oppure la ionizzazione indotta dalla presenza di radiazioni), le popolazioni di elettroni nella banda di valenza e di conduzione cambiano. Se il disturbo non è troppo intenso, i portatori raggiungono uno stato di quasi equilibrio in maniera indipendente. Ciò avviene perché il tempo di rilassamento relativo al passaggio di elettroni fra i livelli energetici all'interno della stessa banda è molto minore di quello relativo al passaggio di elettroni fra le due bande (attraverso il gap).

[7] Un esempio di approccio alternativo risolvere il sistema rispetto alle variabili (ψ, Φ_n, Φ_p), con $\Phi_n = n_i e^{\varphi_n - \psi}$ e $\Phi_p = n_i e^{\phi - \varphi_p}$. In letteratura Φ_n e Φ_p sono dette variabili di Slotboom.

Di conseguenza gli elettroni in banda di valenza e in banda di conduzione hanno due distinti potenziali elettrochimici (a cui sono associate le energie E_{fn} e E_{fp}) e se considerati separatamente seguono la distribuzione di Fermi Dirac

$$f(E) = \frac{1}{1 + e^{\frac{E-\mu}{kT}}}$$

dove μ indica il potenziale elettrochimico associato.

Possiamo quindi utilizzare i modelli[8] e le ipotesi normalmente usate nel caso di silicio a temperatura ambiente in condizioni di equilibrio, sostituendo E_f con E_{fn} e E_{fp}:

$$n = \int_{E_c}^{\infty} D(E) f(E)\, \mathrm{d}E \simeq N_c e^{-\frac{E_c - E_{fn}}{kT}}$$

$$p = \int_{0}^{E_v} D(E)(1 - f(E))\, \mathrm{d}E \simeq N_v e^{-\frac{E_{fp} - E_v}{kT}}$$

In presenza di un potenziale elettrostatico ψ che si somma al piegamento d diventano

$$n = N_c e^{-\frac{E_c - E_{fn}}{kT} + \frac{q\psi}{kT}} = n_i e^{-\frac{E_c - E_{fn}}{kT} + ln\frac{N_c}{n_i} + \frac{q\psi}{kT}}$$

$$p = N_v e^{-\frac{E_{fp} - E_v}{kT} - \frac{q\psi}{kT}} = n_i e^{\frac{E_v - E_{fp}}{kT} + ln\frac{N_v}{n_i} - \frac{q\psi}{kT}}$$

Poniamo

$$\varphi = \frac{q\psi}{kT} = \frac{\psi}{V_t}$$

$$\varphi_n = \frac{E_c - E_{fn}}{kT} - ln\frac{N_c}{n_i}$$

$$\varphi_p = \frac{E_v - E_{fp}}{kT} + ln\frac{N_v}{n_i}$$

da cui si ricava

$$n = n_i e^{\varphi - \varphi_n} \tag{4.17}$$

$$p = n_i e^{\varphi_p - \varphi} \tag{4.18}$$

Sostituendo (4.17) e (4.18) in (4.5) e (4.6) si ricava

$$\overrightarrow{J}_n = q\mu_n(V_t \nabla n - n\nabla\psi) = qV_t\mu_n n_i e^{\varphi - \varphi_n}(\nabla(\varphi - \varphi_n) - \nabla\varphi)$$

$$\overrightarrow{J}_p = q\mu_p(-V_t \nabla p - p\nabla\psi) = qV_t\mu_p n_i e^{\varphi_p - \varphi}(-\nabla(\varphi_p - \varphi) - \nabla\varphi)$$

[8] Dove $D(E)$ indica la densità di volume e di energia degli stati disponibili totali, che incorpora le caratteristiche del cristallo.

e quindi

$$\overrightarrow{J}_n = -kT\mu_n n \nabla \varphi_n \tag{4.19}$$

$$\overrightarrow{J}_p = -kT\mu_p p \nabla \varphi_p \tag{4.20}$$

da cui si ottiene

$$-\frac{1}{q}\nabla \cdot \overrightarrow{J}_n = +\frac{kT}{q}\mu_n (n\,\nabla^2\varphi_n + \nabla\varphi_n \cdot \nabla n)$$

$$+\frac{1}{q}\nabla \cdot \overrightarrow{J}_p = -\frac{kT}{q}\mu_p (p\,\nabla^2\varphi_p + \nabla\varphi_p \cdot \nabla p)$$

Imponiamo le condizioni di regime[9] sulle equazioni di continuità (4.7) e (4.8)

$$V_t \mu_n n_i e^{\varphi-\varphi_n}(\nabla^2\varphi_n + \nabla\varphi_n \cdot (\nabla\varphi - \nabla\varphi_n)) = G - R$$

$$-V_t \mu_p n_i e^{\varphi_p-\varphi}(\nabla^2\varphi_p + \nabla\varphi_p \cdot (\nabla\varphi_p - \nabla\varphi)) = G - R$$

aggiungendo la legge di Poisson e le relazioni di generazione-ricombinazione considerando (4.17) e (4.18) si ricava infine

$$V_t\,\nabla^2\varphi = -\frac{q}{\varepsilon}\left(N_d - N_a + n_i e^{\varphi_p-\varphi} - n_i e^{\varphi-\varphi_n}\right) \tag{4.21}$$

$$V_t \mu_n n_i e^{\varphi-\varphi_n}(\nabla^2\varphi_n + \nabla\varphi_n \cdot \nabla\varphi - \nabla\varphi_n \cdot \nabla\varphi_n) = G - R \tag{4.22}$$

$$V_t \mu_p n_i e^{\varphi_p-\varphi}(\nabla^2\varphi_p\ + \nabla\varphi_p \cdot \nabla\varphi_p\ - \nabla\varphi_p \cdot \nabla\varphi) = R - G \tag{4.23}$$

$$G - R = -\frac{n_i}{\tau}e^{\varphi_p-\varphi_n} \tag{4.24}$$

4.4.1 Mappa di Gummel

Una volta definito il problema, scelta una famiglia di elementi finiti e imposte le condizioni al contorno si può cercare la soluzione con un metodo iterativo non lineare come il metodo di Newton.

Il metodo di Netwon presenta tuttavia due criticità:

- necessità di una soluzione di partenza abbastanza vicina a quella finale;
- calcolo della matrice jacobiana del sistema, pesante dal punto di vista computazionale nel caso di sistemi 3D.

[9] $\frac{\partial n}{\partial t} = \frac{\partial p}{\partial t} = 0$

L'algoritmo di Gummel, detto anche Gummel map, è una tecnica che consente di affrontare entrambi i problemi. Si risolvono separatamente e iterativamente le tre equazioni disaccoppiate, prima quella del potenziale elettrico e successivamente le altre due, riducendo la complessità della singola singola iterazione. La mappa di Gummel non consente di approssimare con elevata precisione la soluzione ottima. La sua popolarità è dovuta all'ampio raggio e all'elevata velocità di convergenza: con poche iterazioni è possibile trovare una buona soluzione di partenza su cui impiegare il metodo di Netwon, più accurato.

$$\varphi_{k+1} \rightarrow (\varphi_{pk}, \varphi_{nk}, \varphi_k)$$
$$\varphi_{nk+1} \rightarrow (\varphi_{pk}, \varphi_{nk}, \varphi_{k+1})$$
$$\varphi_{pk+1} \rightarrow (\varphi_{pk}, \varphi_{nk+1}, \varphi_{k+1})$$

Notare che in letteratura l'equazione del potenziale elettrico può essere risolta in versione non lineare ($\varphi_{k+1} \rightarrow (\varphi_{pk}, \varphi_{nk}, \varphi_{k+1})$). Se il contributo di generazione ricombinazione è piccolo conviene comunque risolvere l'equazione di Poisson in versione lineare ed ottenere la soluzione più velocemente[86].

4.4.2 Scelta delle variabili

La scelta della terna di variabili adimensionali (φ, φ_p, φ_n) porta a dei vantaggi rispetto alla terna (ψ, p, n).

Storicamente esisteva un importante fattore: il range necessario per rappresentare le concentrazioni dei portatori spazia su molte decadi e se si ricorre ad aritmetica fixed point ciò porta ad un pessimo compromesso in termini di memoria e precisione. Di conseguenza, sfruttare la relazione esponenziale permette di aumentare l'efficienza complessiva della risoluzione.

Vediamo brevemente come si impongono le condizioni al contorno. Prendiamo il caso di un cristallo estrinseco n sotto l'ipotesi che la concentrazione di maggioritari sia pari al drogaggio

$$n = N_d = n_i e^{\varphi - \varphi_n}$$
$$p = \frac{n_i^2}{N_d} = n_i e^{\varphi_p - \varphi}$$

da cui si ricava

$$\varphi_n = \varphi - \ln \frac{N_d}{n_i}$$
$$\varphi_p = \varphi - \ln \frac{N_d}{n_i}$$

Prendiamo il caso speculare di un cristallo p sotto le medesime ipotesi

$$n = \frac{n_i^2}{N_a} = n_i e^{\varphi - \varphi_n}$$

$$p = N_a = n_i e^{\varphi_p - \varphi}$$

da cui si ricava

$$\varphi_n = \varphi + \ln \frac{N_a}{n_i}$$

$$\varphi_p = \varphi + \ln \frac{N_a}{n_i}$$

La concentrazione quindi può essere imposta mediante una condizione di Dirichlet, esattamente come si farebbe con la terna (ψ, p, n).

Imporre invece l'assenza di passaggio di corrente attraverso una superficie diventa una condizione di Neumann

$$\overrightarrow{J}_n = 0 \rightarrow \varphi_n = 0$$

$$\overrightarrow{J}_p = 0 \rightarrow \varphi_p = 0$$

mentre con la terna (ψ, p, n) diventa una condizione di Robin

$$\overrightarrow{J}_n = 0 \rightarrow D_n \nabla n - \mu_n n \nabla \psi = 0$$

$$\overrightarrow{J}_p = 0 \rightarrow -D_p \nabla p - \mu_p p \nabla \psi = 0$$

4.4.3 Schema di Slotboom

In alternativa si può usare la schematizzazione di Slotboom partendo dalle (4.17) e (4.18), poniamo:

$$u_n = N_c e^{-\varphi_n}$$

$$u_p = N_v e^{+\varphi_p}$$

da cui si ricava

$$n = N_c e^{\varphi - \varphi_n} = u_n e^{+\varphi}$$

$$p = N_v e^{\varphi_p - \varphi} = u_p e^{-\varphi}$$

Dalle espressioni per la corrente (4.5) e (4.6)

$$\overrightarrow{J}_n = q\mu_n(V_t u_n e^{+\varphi} \nabla \varphi + V_t e^{+\varphi} \nabla u_n - n \nabla \psi) = q V_t \mu_n e^{+\varphi} \nabla u_n$$

$$\overrightarrow{J}_p = q\mu_p(-V_t u_p e^{-\varphi} \nabla \varphi + V_t e^{-\varphi} \nabla u_p + p \nabla \psi) = q V_t \mu_p e^{-\varphi} \nabla u_p$$

Usiamo le equazioni di continuità (4.7) (4.8) e imponiamo le condizioni a regime

$$-V_t \nabla \cdot (\mu_n (e^{+\varphi} \nabla u_n)) = G - R$$
$$-V_t \nabla \cdot (\mu_p (e^{-\varphi} \nabla u_p)) = G - R$$

Si ricava

$$-V_t \mu_n N_c \nabla \cdot (e^{+\varphi} \nabla e^{-\varphi_n}) = G - R$$
$$-V_t \mu_p N_v \nabla \cdot (e^{-\varphi} \nabla e^{+\varphi_p}) = G - R$$

che sono equivalenti alle equazioni già trovate (4.22) e (4.23)

Capitolo 5

Implementazione e risultati

Dopo aver definito le equazioni e le forme variazionali da risolvere rimane da esporre come sono state implementate. In questo capitolo saranno mostrati alcuni esempi di applicazione del framework offerto da FEniCS, descrivendo il flusso di lavoro di un caso ideale.

Data la semplicità di lettura del codice Python e dalla chiarezza delle funzionalità della libreria, saranno mostrate e commentate sezioni del programma realmente implementate, tranne nel caso in cui il significato non sia sufficientemente intuitivo. Per completezza, sarà mostrato brevemente l'uso di un CAD per il disegno di un layout.

5.1 Concetti generali

Fra i fattori che hanno portato alla scelta del framework FEniCS vi è senza dubbio la possibilità di compilare le espressioni e gli algoritmi a partire da una descrizione ad alto livello. Il fatto di poter fornire dati e espressioni al run time permette sia di testare in ambiente interattivo il corretto funzionamento del codice sia di separare in maniera efficace i dati dal codice.

5.1.1 Compilazione automatica

La gestione della compilazione è trasparente all'utente. La prima volta che il codice viene eseguito viene creato un file compilato da eseguire. Le successive esecuzioni saranno molto più veloci rispetto alla prima, dato che non saranno necessarie nuove compilazioni. Questo approccio facilita e velocizza la fase di testing, ma consente anche di ottimizzare notevolmente e senza sforzo sezioni di codice eseguite più volte, come

succede nelle iterazioni per la risoluzione di sistemi non lineari o per la descrizione di transitori nel tempo.

Un esempio particolarmente interessante sono le cosidette "espressioni", termine che in FEniCS indica sezioni di codice scritti in C++ e compilate su richiesta. Le espressioni sono, ai fini dell'implementazione, delle semplici stringhe in Python. Il codice consente di descrivere un campo scalare sul dominio. Per descrivere un campo vettoriale si possono unire più espressioni. Questo consente di esprimere le condizioni di Neumann in funzione delle coordinate con semplicità. A differenza di altri progetti simili come il già citato pacchetto weave di scipy che richiedono una sintassi complessa per eseguire lo stesso compito, FEniCS fornisce il vettore della posizione come array composto da tre coordinate. Per esempio un semplice campo scalare definito come $sin(x) + cos(y) + z^2$ è traducibile come:

```
Expression('sin(x[0]) + cos(x[1]) + x[2]*x[2]')
```

Con una sintassi più facile da imparare e da leggere il rischio di errori cala.

Grazie alle espressioni, la distribuzione di portatori generati dal passaggio di una particella ionizzante può essere inserita fra i dati iniziali della simulazione in maniera flessibile sottoforma di stringa, sia in 2D sia in 3D. Un altro possibile uso delle espressioni è nella descrizione dei drograggi all'interno dispositivo o di eventuali variazioni di parametri come temperatura, permeabilità dielettrica o mobilità.

5.1.2 Input dei dati

Solitamente i framework per simulazioni numeriche non dispongono di strumenti per il trattamento delle grandezze fisiche. Il motivo è che tutti i problemi variazionali possono essere espressi in forma adimensionale. Se il sistema è normalizzato in maniera scorretta l'esecuzione dell'algoritmo potrebbe portare ad errori calcolo o quanto meno ad una riduzione della precisione. Mentre in linguaggi come Python l'interprete gestisce automaticamente questi casi particolari, le librerie per calcolo scientifico non possono permettersi un overhead così intenso e di conseguenza si limitano a segnalare l'eventuale errore e di conseguenza il compito di scegliere una normalizzazione corretta che riduca la perdita di precisione è lasciata all'utente.

Tuttavia, poter esprimere i parametri come quantità dimensionali con unità fisiche significative ha il grande vantaggio di rendere l'utilizzo più facile e meno prono ad errori. I dati e i parametri di simulazione sono contenuti in file di configurazione. Si sarebbe potuto impiegare

anche il formato XML, che è lo standard più diffuso per la memorizzazione di dati in formato human-redable in sistemi multipiattaforma, ma si è preferito impiegare lo standard JSON perché ritenuto più leggibile.

Per esempio, una struttura gerarchica di dati in XML come

```
<simulation>
    <tolerance>
        1.e-4
    </tolerance>
    <steps>
        10
    </steps>
</simulation>
```

può essere scritta in maniera più compatta e leggibile in JSON

```
{
    "simulation" : {
        "tolerance" : 1.e-4,
        "steps" : 10
    }
}
```

Per funzionare correttamente, il modulo per la conversione delle unità di misura richiede che le quantità siano memorizzate come stringhe con l'unità di misura. In compenso, il sistema gestisce correttamente anche unità non SI e anche le principali costanti fisiche. Per esempio, le proprietà fisiche del silicio sono memorizzate in questa forma:

```
{
    "silicon" : {
        "T" : "300 K",
        "E_gap" : "1.08 eV",
        "permittivity" : "11.8 eps0",
        "Mu_n" : "1350 cm**2 / V / s",
        "Mu_p" : "495 cm**2 / V / s",
    }
}
```

La permittività è espressa in funzione della costante dielettrica del vuoto, mentre sia l'energia del gap sia le mobilità dei portatori sono espresse in unità di comodo.

5.2 Creazione e definizione del mesh

Il primo passo è la definizione di un mesh. Esistono due modi per creare un mesh in FEniCS:

- attraverso le funzioni di FEniCS;
- importando un file creato con un CAD esterno.

Le funzioni built-in forniscono delle forme semplici, come rettangoli e parallelepipedi, d'uso intuitivo. Per esempio, un parallelepipedo può essere caratterizzato in base alle coordinate di due angoli opposti. Per definire il mesh, è necessario aggiungere in quanti elementi è diviso lungo i tre assi. Di default la griglia è regolare, ma può essere manipolata per ottenere meshing non regolari complessi.

```
BoxMesh(Point(x0,y0,z0), Point(x1,y1,z1), nx, ny, nz)
```

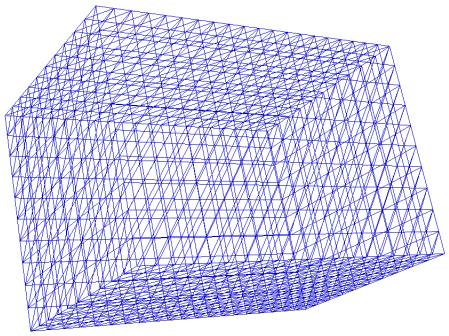


Figura 5.1: Esempio di mesh built-in con FEniCS

Anche se è possibile creare mesh più complessi componendo figure elementari, nella pratica conviene impiegare creare un file con un CAD e successivamente importarlo. FEniCS può importare una ampia varietà di formati, ma in questo progetto si è impiegato il CAD open source GMSH.

5.2.1 Uso di GMSH

GMSH è un CAD per la generazione di mesh 2D e 3D, con l'abilità di integrare risolutori esterni. GMSH è installabile sui principali si-

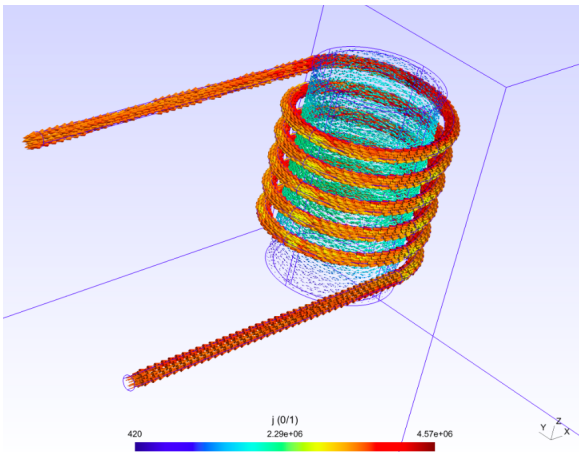


Figura 5.2: Simulazione di un induttore all'interno dell'ambiente GMSH. Tratto da [92]

stemi operativi ed è perfino disponibile (con funzionalità ridotte) per dispositivi mobile.

In GMSH è possibile impiegare un linguaggio di scripting per descrivere gli enti geometrici più semplici e unirli in modo da rappresentare enti geometrici superiori. Tutte le funzioni possibili attraverso scripts possono essere eseguite con l'impiego di una interfaccia grafica interattiva. Le linee rette per esempio sono rappresentate come unione di due punti, le superfici dalle linee che ne compongono il contorno e i volumi attraverso le superfici che le delimitano.

```
Point(1) = {0, 0, 0};
Point(2) = {0, 1, 0};
Point(3) = {1, 1, 0};
Line(1) = {1, 2};
Line(2) = {2, 3};
Line(4) = {3, 1};
```

Una volta fornito un identificativo interno a tutte le entità geometriche è possibile definire degli insiemi di significato fisico (dette *physical regions*) su cui andare a fissare le condizioni al contorno, come le linee che costituiscono il contorno di una superficie:

```
Physical Line(1) = {1};
Physical Line(2) = {2};
```

Allo stesso modo si possono definire enti geometrici superiori come superfici o volumi.

```
Physical Surface(1) = {1};
```

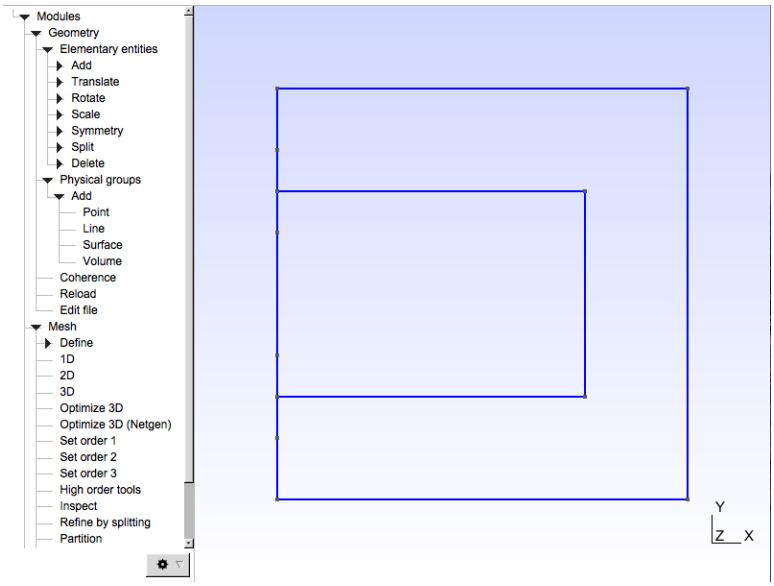


Figura 5.3: Esempio semplice di layout in GMSH

```
Physical Volume(1) = {1};
```

Una volta definiti i contorni e le zone in cui è diviso il dominio, è importante che il codice identificativo impiegato nel mesh sia il medesimo usato all'interno di FEniCS. Per motivi legati al funzionamento interno di FEniCS, i vari enti fisici devono essere numerati a partire da 1 nel file del mesh generato da GMSH.

Il file generato da GMSH (o da qualiasi altro CAD) non può essere impiegato direttamente da FEniCS. FEniCS sfrutta un formato interno basato su XML e divide il mesh vero e proprio, i contorni e i sottodominii in tre file differenti. All'interno del pacchetto è disponibile una funzionalità da linea di comando che consente di estrarre queste informazioni a partire dal file mesh generato da GMSH.

Una volta completata l'operazione di conversione, l'importazione è una operazione semplice.

```
mesh = Mesh('mesh.xml')
```

5.2.2 Spazi di funzioni

Il mesh è solo una delle componenti della discretizzazione del problema. Per completare la descrizione è necessario indicare quale famiglia di funzioni deve essere impiegata ed, eventualmente, con quale grado. Per esempio, per usare funzioni continue di Lagrange al primo ordine è sufficiente scrivere:

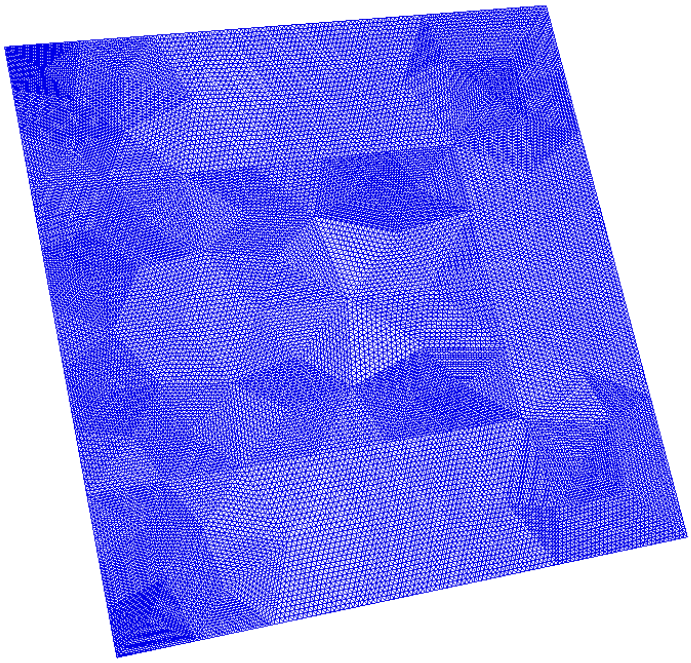


Figura 5.4: Mesh costruito in GMSH e importato in FEniCS.

```
V = FunctionSpace(mesh, 'Lagrange', 1)
```

Naturalmente esistono possibilità più esotiche, come spazi di Lagrange discontinui (indicati sotto il nome di Discontinous Galerkin, DG) oppure spazi di funzioni costanti:

```
V = FunctionSpace(mesh, 'DG', 1)
V = FunctionSpace(mesh, 'R', 0)
```

Si possono unire più spazi di funzioni, in modo da ottenere famiglie miste o campi complessi

```
V_l = FunctionSpace(mesh, 'DG', 1)
V_r = FunctionSpace(mesh, 'R', 0)
W_lr = MixedFunctionSpace([V_l, V_r])
```

oppure funzioni vettoriali

```
V_x = FunctionSpace(mesh, 'DG', 1)
V_y = FunctionSpace(mesh, 'DG', 1)
V_z = FunctionSpace(mesh, 'DG', 1)
W_xyz = MixedFunctionSpace([V_x, V_y, V_z])
```

Anche se è non una operazione banale, è possibile definire spazi di funzioni custom. Una volta definito lo spazio di lavoro, è semplice creare una funzione.

```
f = Function(V)
```

Le funzioni sono rappresentate attraverso vettori che possono essere manipolati con pacchetti di calcolo matriciale esterni, per esempio Numpy. FEniCS mette a disposizione alcuni metodo di uso comune, come l'interpolazione di una funzione in un altro spazio di funzioni:

```
f_1 = Function(V_1)
f_2 interpolate(f_1, V_2)
```

Le funzioni di prova e di test possiedono una notazione apposita, dato che non hanno un valore fissato ma rappresentano solo delle variabili utili a definire il problema variazionale:

```
u = TrialFunction(V)
v = TestFunction(V)
```

5.3 Gestione delle condizioni al contorno

Per gestire una condizione al contorno il primo passo è definire il confine. Per farlo, deve essere possibile identificare una zona del mesh. Per condizioni particolarmente semplici si può definire un contorno come una funzione booleana. Le coordinate sono disponibili sotto forma di vettore (x, y, z), accessibile come un array, esattamente come visto nelle espressioni.

```
def left_face(x):
    return x[0] == 0
```

Dato che potrebbero esserci errori nelle uguaglianze fra numeri reali a causa della precisione limitata delle coordinate nel mesh, conviene effetturare il confronto con una certa tolleranza. Per semplificare la gestione di questi casi, esiste un argomento che permette di distinguere con sicurezza se un punto del mesh è al bordo o meno.

```
def right_top_corner(x, on_boundary):
    tol = 1E-15
    return abs(x[0] - 1) < tol and abs(x[1] - 1) < tol
```

Una volta definiti i contorni, resta da imporre le condizioni. Come visto nel capitolo precedente, le condizioni di Neumann entrano direttamente nella definizione del problema variazionale, mentre le condizioni di

Dirichlet definiscono l'insieme delle funzioni di prova. Per comodità di implementazione FEniCS gestisce le condizioni di Dirichlet alla risoluzione del problema variazionale invece che alla dichiarazione dello spazio di funzione di prova, come teoricamente prescrive il metodo degli elementi finiti. Posto di aver definito uno spazio di funzioni, un valore numerico e una funzione booleana che rappresenta il contorno, è semplice definire una condizione di Dirichlet:

```
bc = DirichletBC(V, 0, left_boundary)
```

In alternativa, si può condensare la sintassi con l'impiego di una espressione implicita, per esempio:

```
bc = DirichletBC(V, 0, 'x[0] < tol and on_boundary')
```

Naturalmente più condizioni di Dirichlet possono essere unite in array di condizioni al contorno:

```
bcs = [
    dolfin.DirichletBC(V, 0, left_boundary),
    dolfin.DirichletBC(V, 1, right_boundary),
]
```

dove è stato posto 0 come valore per la condizione di Dirichlet a sinistra e 1 a destra.

Una tecnica più complessa ma simile deve essere impiegata per la definizione di sottodomini. È necessario incapsulare l'espressione booleana da testare all'interno di una classe

```
class left_region(SubDomain):
    def inside(self, x):
        return x[0] <= 0.5
```

In secondo luogo è necessario definire una funzione speciale apposita per la definizione dei sottodominii sul mesh:

```
subdomains = CellFunction('size_t', mesh, mesh_dimension)
```

Infine, si può identificare il sottodominio con un identificativo numerico:

```
left_region().mark(subdomains, 0)
```

In alternativa, sottodominii e condizioni al contorno possono essere definiti in GMSH come *physical regions* e successivamente importati in FEniCS:

```
mesh = Mesh('mesh.xml')
subdomains = MeshFunction('size_t', mesh, \
    'mesh_physical_region.xml')
boundaries = MeshFunction('size_t', mesh, \
    'mesh_facet_region.xml')
```

L'unico caveat è che l'identificativo usato in GMSH deve essere un numero intero maggiore di 0 e che deve corrispondere a quelli impiegati in FEniCS.

5.4 Definizione del problema variazionale

Dopo aver definito la discretizzazione del problema e le condizioni al contorno, è necessario impostare il problema variazionale da risolvere. Per esempio, riprendendo il caso generale di Poisson:

$$\int_\Omega \nabla \mathrm{u}_h \cdot \nabla \mathrm{v}_h \, \mathrm{d}\omega = \int_\Omega f \mathrm{v}_h \, \mathrm{d}\omega - \int_{\partial\Omega_N} g \, \mathrm{v}_h \, \mathrm{d}s$$

Posto che f e g siano due espressioni, l'equazione in Python diventa:

```
eqn = inner(grad(u), grad(v)) == f * v * dx - g * v * ds
```

dove con *inner*() indica il prodotto fra vettori (dato che evidentemente ∇u e ∇v sono due funzioni vettoriali) mentre dx e ds indicano la variabile d'integrazione. Il risultato deve essere salvato in una funzione, per esempio φ, e bisogna fornire una lista di condizioni al contorno di Dirichlet. Per risolvere il problema con le impostazioni di default di FEniCS è necessario un solo passo:

```
phi = Function(V)
solve(eqn, phi, bcs)
```

Tuttavia, il framework mette a disposizione anche le opzioni necessarie per una regolazione più fine della risoluzione o per separare la risoluzione nei vari passaggi. Posto di aver separato i membri dell'equazione nella forma bilineare a e nella forma lineare L:

```
problem = LinearVariationalProblem(a, L, phi, bcs)
solver = LinearVariationalSolver(problem)
solver.solve()
```

Con questa tecnica, si possono impostare parametri come il precondizionamento delle matrici sul un risolutore lineare oppure il numero massimo di iterazioni o la tolleranza assoluta su un risolutore iterativo.

Il problema variazionale può essere espresso in modo da variare a seconda del sottodominio, per esempio se nel dominio 1 si ha una sorgente rappresentata dall'espressione f_1 e nel dominio 2 si ha una sorgente rappresentata dall'espressione f_2 l'equazione di Poisson diventa:

```
eqn = inner(grad(u), grad(v)) == \
    f_1 * v * dx(1) + f_2 * v * dx(2)
```

Come è stato anticipato nel capitolo precedente, il framework FEniCS consente l'automatizzazione dell'adattamento del mesh, oltre che della risoluzione. Da un punto di vista pratico, è necessario definire un criterio di valutazione dell'errore e una tolleranza massima. Dato che il criterio è arbitrario si parla di adattabilità goal oriented. Per l'utente, l'operazione è semplificata dalla interfaccia FEniCS:

```
tolerance = 1.e-2
M = u*dx
solve(F == 0, u, bcs, tol=tolerance, M=M)
```

dove F è un problema rispetto alla variabile u e *bcs* è una lista di condizioni di Dirichlet da applicare. La matrice M rappresenta la grandezza rispetto a cui minimizzare l'errore. Si possono scegliere anche criteri più complessi, come il flusso sulle superfici del dominio.

5.4.1 Simulazione a regime

FEniCS consente di risolvere anche i problemi non lineari mediante l'uso di metodi built in. Riprendiamo per esempio l'equazione non lineare per il potenziale 4.21, ricordando che φ, φ_n e φ_p sono grandezze adimensionale e che φ è pari al potenziale elettrostatico scalato rispetto alla tensione V_t:

$$V_t \nabla^2 \varphi = -\frac{q}{\varepsilon} \left(N_d - N_a + n_i e^{\varphi_p - \varphi} - n_i e^{\varphi - \varphi_n} \right)$$

Per mostrare un esempio in una sola variabile, ipotiziamo di essere all'equilibrio. In assenza di corrente si ha $\varphi_n = 0$ e $\varphi_p = 0$. L'equazione può essere riscritta come funzione dell'energia di gap e del livello di Fermi, in questa forma:

$$V_t \nabla^2 \varphi = -\frac{q}{\varepsilon} \left(N_d - N_a + N_v e^{-\frac{E_f}{kT}} e^{-\varphi} - N_c e^{\frac{E_f - E_{gap}}{kT}} e^{\varphi} \right)$$

Il sistema può essere risolto direttamente in FEniCS, se espresso come forma variazionale non lineare. Moltiplichiamo per la funzione di test:

$$V_t \nabla \varphi \cdot \nabla \mathrm{u} = \frac{q}{\varepsilon} \left(N_d - N_a + N_v e^{-\frac{E_f}{kT}} e^{-\varphi} - N_c e^{\frac{E_f - E_{gap}}{kT}} e^{\varphi} \right) \mathrm{u}$$

Si porta il problema nella forma $F = 0$. Nella pratica, la densità di carica fissa derivante dalla differenza fra N_a e N_d sarà inclusa in una funzione unica, dipendente dalle coordinate spaziali, che sarà chiamata *ions*. Il mesh è scalato rispetto ad un riferimento di lunghezza L e di coseguenza il termine legato a $\nabla^2 \varphi$ deve essere diviso per L^2. Sotto queste ipotesi la forma variazionale diventa:

```
F = (Vt / (L * L)) * inner(grad(phi), grad(v)) * dx - \
    (q / permittivity) * ( \
        ions \
        + Nv * exp(- E_f / kT) * exp(-phi) \
        - Nc * exp(E_f / kT - E_gap / kT) * exp(+phi) \
    ) * v * dx
```

A questo punto, per la risoluzione è necessario identificare lo jacobiano della matrice associata al problema. Questo passo può essere eseguito a mano, ma FEniCS consente di automatizzarlo:

```
dphi = TrialFunction(V)
J = derivative(F, phi, dphi)
problem = NonlinearVariationalProblem(F, phi, bcs, J)
solver = NonlinearVariationalSolver(problem)
```

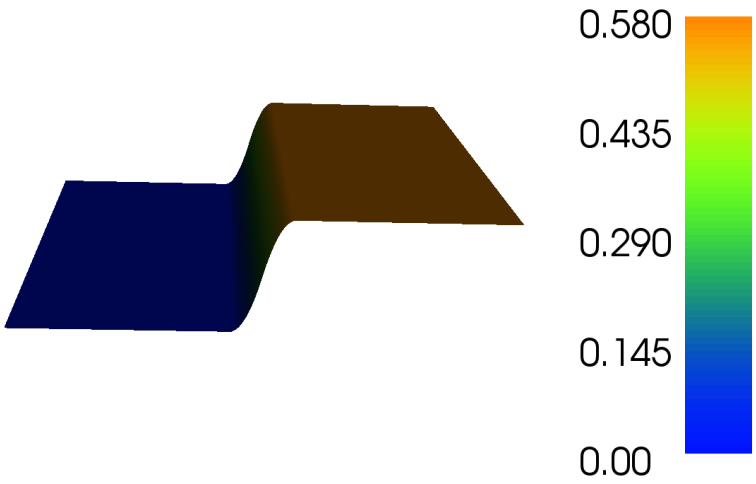


Figura 5.5: Andamento del potenziale in presenza di una giunzione brusca all'equilibrio, con Na = 10^{15}cm^{-3} e Nd = 10^{15}cm^{-3}.

Per implementare l'algoritmo della mappa di Gummel ed ottenere una soluzione approssimata si ripete lo stesso procedimento per tutte e tre le equazioni, eventualmente sostituendo il termine di generazione ricombinazione 4.24 all'interno delle equazioni 4.21, 4.22 e 4.23:

$$V_t \nabla^2 \varphi = -\frac{q}{\varepsilon}\left(N_d - N_a + n_i e^{\varphi_p - \varphi} - n_i e^{\varphi - \varphi_n}\right)$$

$$V_t \mu_n n_i e^{\varphi - \varphi_n}(\nabla^2 \varphi_n + \nabla \varphi_n \cdot \nabla \varphi - \nabla \varphi_n \cdot \nabla \varphi_n) + \frac{n_i}{\tau} e^{\varphi_p - \varphi_n}$$

$$V_t \mu_p n_i e^{\varphi_p - \varphi}(\nabla^2 \varphi_p + \nabla \varphi_p \cdot \nabla \varphi_p - \nabla \varphi_p \cdot \nabla \varphi) - \frac{n_i}{\tau} e^{\varphi_p - \varphi_n}$$

Il termine legato alla tensione è discretizzato come già mostrato:

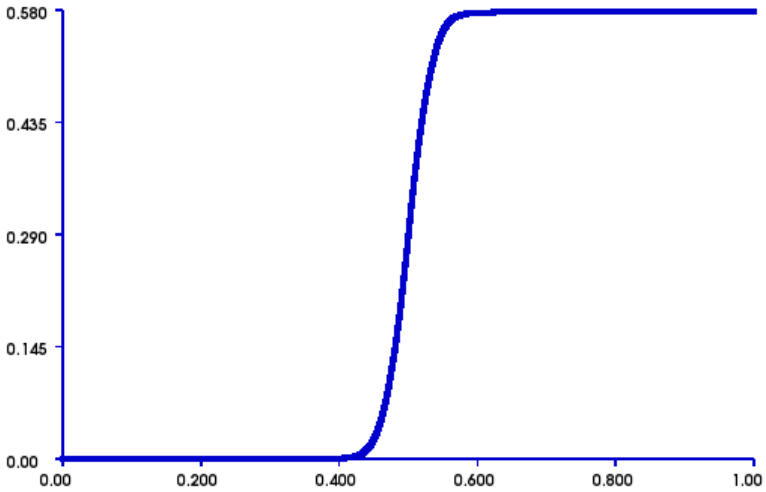


Figura 5.6: Sezione del potenziale elettrico [V]. Mesh normalizzato rispetto a 10μm

```
F = (Vt / (L * L)) * inner(grad(phi), grad(v)) * dx - \
    (q / permittivity) * ( \
        ions \
        + Ni * exp(+phi_p) * exp(-phi) \
        - Ni * exp(-phi_n) * exp(+phi) \
    ) * v * dx
```

Per i quasi livelli si applica una procedura simile:

```
F_n = (Vt * Mu_n * Ni / (L * L)) * \
```

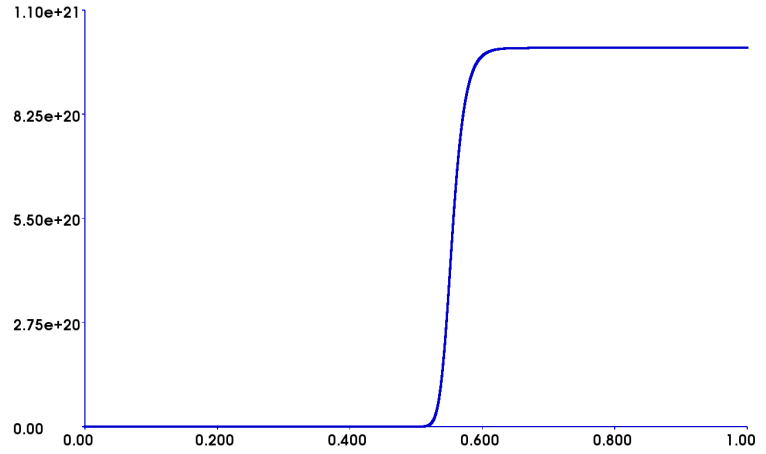


Figura 5.7: Sezione degli elettroni [m^{-3}]. Mesh normalizzato rispetto a 10μm

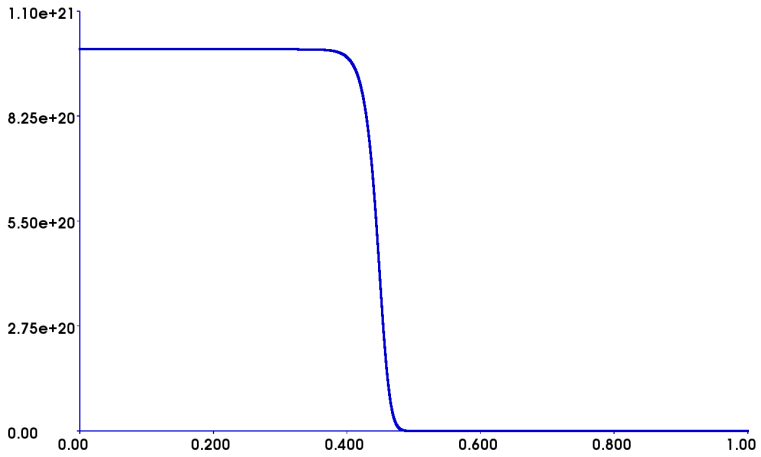


Figura 5.8: Sezione delle lacune [m^{-3}]. Mesh normalizzato rispetto a 10μm

```
    exp(+phi) * exp(-phi_n) * ( \
        - inner(grad(phi_n), grad(v_n)) \
        + inner(grad(phi_n), grad(phi)) * v_n \
        - inner(grad(phi_n), grad(phi_n)) * v_n \
    ) * dx \
    + Ni / lifetime * exp(phi_p) * exp(-phi_n) * v_n * dx
F_p = (Vt * Mu_p * Ni / (L * L)) * \
    exp(+phi_p) * exp(-phi) * ( \
        - inner(grad(phi_p), grad(v_p)) \
        + inner(grad(phi_p), grad(phi_p)) * v_p \
        - inner(grad(phi_p), grad(phi)) * v_p \
    ) * dx + \
    - Ni / lifetime * exp(phi_p) * exp(-phi_n) * v_p * dx
```

Risolvendo separatamente e iterativamente i tre problemi si ricava una soluzione sufficientemente vicina a quella finale. Per applicare infine Netwon è sufficiente risolvere il problema

```
F + F_n + F_p == 0
```

a patto che le tre variabili facciano parte del medesimo spazio misto di funzioni.

5.4.2 Simulazione nel tempo

Come è stato già fatto notare nel capitolo precedente, la simulazione in transitorio richiede solo la risoluzione di sistemi lineari. In questo

caso, mostreremo solo come si risolve solo una delle equazioni di uno dei portatori, dato che si tratta di tre equazioni simili.

Riprendiamo l'equazione di drift-diffusion per le lacune 4.16:

$$\frac{p_{k+1} - p_k}{\Delta t} = \mu_p(V_t \nabla^2 p_{k+\frac{1}{2}} + \nabla p_{k+\frac{1}{2}} \cdot \nabla \psi_k + p_{k+\frac{1}{2}} \nabla^2 \psi_k) + G - R_k$$

dove era stato posto:

$$p_{i+\frac{1}{2}} = \frac{p_i + p_{i+1}}{2}$$

Anche in questa occasione, si moltiplica per la funzione di test v e si ricava:

```
F_p = (1/dt) * (trial_p - old_p)* v * dx \
    + D_p * inner(grad(trial_p),grad(v)) * dx \
    + Mu_p * dot (grad(phi),grad(p_mid)) * v * dx \
    + Mu_p * p_mid * dot(grad(phi),grad(phi)) * v * dx \
    - (g - r) * v * dx
```

Per risolvere il sistema rispetto al nuovo valore p_{k+1} è necessario impiegare la funzione di prova *trial_p*. Il problema deve essere risolto iterativamente, insieme alle altre equazioni. In alcune condizioni, si può ottimizzare i cicli necessari attraverso una apposita fattorizzazione delle matrici.

Bibliografia

[1] ESA-ESTEC Data Systems Division / Microelectronics Section Noordwijk, The Netherlands "Space engineering, product assurance. Techniques for Radiation Effects. Mitigation in ASICs and FPGAs." ESA-HB-XX-XX-rev.6. 2 December 2011.

[2] Brenner, David J., et al. "Cancer risks attributable to low doses of ionizing radiation: assessing what we really know." Proceedings of the National Academy of Sciences 100.24 (2003): 13761-13766.

[3] Colinge, J-P. Silicon-on-Insulator Technology: Materials to VLSI: Materials to Vlsi. Springer Science and Business Media, 2004.

[4] Geiger–Müller tube, Wikipedia: The Free Encyclopedia. Wikimedia Foundation, `https://en.wikipedia.org/wiki/Geiger\T1\textendashMüller_tube`

[5] Roentgen, Wikipedia: The Free Encyclopedia. Wikimedia Foundation, `https://en.wikipedia.org/wiki/Roentgen_(unit)`

[6] Greco, Carlo, and Suzanne Wolden. "Current status of radiotherapy with proton and light ion beams." Cancer 109.7 (2007): 1227-1238.

[7] Edward Petersen, "Single Event Effects in Aerospace", John Wiley and Sons, 2011

[8] Vazquez-Luque, Aurelio, et al. "Neutron Induced Single Event Upset Dependence on Bias Voltage for CMOS SRAM With BPSG." Nuclear Science, IEEE Transactions on 60.6 (2013): 4692-4696.

[9] Dodd, P.E.; Shaneyfelt, M.R.; Schwank, J.R.; Felix, J.A., "Current and Future Challenges in Radiation Effects on CMOS Electronics," in Nuclear Science, IEEE Transactions on , vol.57, no.4, pp.1747-1763, Aug. 2010 doi: 10.1109/TNS.2010.2042613

[10] Abraham, J; Abreu, P; Aglietta, M; Aguirre, C; Allard, D; Allekotte, I; Allen, J; Allison, P; Alvarez, C; Alvarez-Muñiz, J; "Correlation of the highest-energy cosmic rays with nearby extragalactic objects,Science", Science, vol.318, pp.938-943, 2007 doi: 5852

[11] Wallmark, J.T.; Marcus, S.M., "Minimum Size and Maximum Packing Density of Nonredundant Semiconductor Devices," in Proceedings of the IRE, vol.50, no.3, pp.286-298, March 1962 doi: 10.1109/JRPROC.1962.288321

[12] Binder, D.; Smith, E.C.; Holman, A.B., "Satellite Anomalies from Galactic Cosmic Rays," in Nuclear Science, IEEE Transactions on, vol.22, no.6, pp.2675-2680, Dec. 1975 doi: 10.1109/TNS.1975.4328188

[13] Field, R. William, et al. "Residential radon gas exposure and lung cancer the iowa radon lung cancer study." American Journal of Epidemiology 151.11, pp. 1091-1102, 2000

[14] Darby, Sarah, et al. "Radon in homes and risk of lung cancer: collaborative analysis of individual data from 13 European case-control studies." Bmj 330.7485, pp. 223, 2005

[15] Jean-Luc Autran, Daniela Munteanu "Soft Errors: From Particles to Circuits", CRC Press, pp. 72-73, Feb 25, 2015

[16] Schou, Jørgen. "Transport theory for kinetic emission of secondary electrons from solids." Physical Review B 22.5 (1980): 2141.

[17] Turner, J. E. and Kher, R. K. and Arora, D. and Bisht, J. S. and Neelavathi, V. N. and Vora, R. B. "Quantum Mechanical Calculation of Neutron Stopping Power",Phys. Rev. B,American Physical Society, vol.8, no. 9, pp. 4057–4062, Nov. 1973 doi:10.1103/PhysRevB.8.4057

[18] Allan Johnston, "Reliability and Radiation Effects in Compound Semiconductors", World Scientific, 2010

[19] Barnaby, H. J. "Total-ionizing-dose effects in modern CMOS technologies." Nuclear Science, IEEE Transactions on 53.6 (2006): 3103-3121.

[20] Oldham, Timothy R., and F. B. McLean. "Total ionizing dose effects in MOS oxides and devices." IEEE Transactions on Nuclear Science 50.3 (2003): 483-499.

[21] Moll, Michael, and RD50 Collaboration. "Development of radiation hard sensors for very high luminosity colliders—CERN-RD50 project." Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 511.1 (2003): 97-105.

[22] Atlas experiment, `http://atlas.ch`

[23] Pickel, James C., and James T. Blandford Jr. "Cosmic-ray-induced errors in MOS devices." Nuclear Science, IEEE Transactions on 27.2 (1980): 1006-1015.

[24] Massengill, L. W., et al. "Effects of process parameter distributions and ion strike locations on SEU cross-section data [CMOS SRAMs]." Nuclear Science, IEEE Transactions on 40.6 (1993): 1804-1811.

[25] Petersen, E. L., et al. "Calculation of cosmic-ray induced soft upsets and scaling in VLSI devices." Nuclear Science, IEEE Transactions on 29.6 (1982): 2055-2063.

[26] Dodd, P. E., and F. W. Sexton. "Critical charge concepts for CMOS SRAMs." Nuclear Science, IEEE Transactions on 42.6 (1995): 1764-1771.

[27] Jean-Luc Autran, Daniela Munteanu, "Soft Errors: From Particles to Circuits", CRC Press, 2015

[28] May, Timothy C., and Murray H. Woods. "A new physical mechanism for soft errors in dynamic memories." Reliability Physics Symposium, 1978. 16th Annual. IEEE, 1978.

[29] Johnston, Allan H. "Scaling and technology issues for soft error rates." Proc. of the 4th Annual Conf. on Reliability. 2000.

[30] Diniz, Josté, et al. "Proton radiation hardening of silicon oxynitride gate nMOSFETs formed by nitrogen implantation into silicon prior to oxidation." Radiation and Its Effects on Components and Systems, 2001. 6th European Conference on. IEEE, 2001.

[31] Mrstik, B. J., et al. "Hole and electron trapping in ion implanted thermal oxides and SIMOX." Nuclear Science, IEEE Transactions on 47.6 (2000): 2189-2195.

[32] Nishioka, Yasushiro, et al. "Radiation hardened micron and submicron MOSFETs containing fluorinated oxides." Nuclear Science, IEEE Transactions on 36.6 (1989): 2116-2123.

[33] Kato, Masataka, Kikuo Watanabe, and Takeaki Okabe. "Radiation effects on ion-implanted silicon-dioxide films." Nuclear Science, IEEE Transactions on 36.6 (1989): 2199-2204.

[34] Schrankler, J. W., et al. "CMOS scaling implications for total dose radiation." Nuclear Science, IEEE Transactions on 32.6 (1985): 3988-3990.

[35] Shaneyfelt, M. R., et al. "Effects of device scaling and geometry on MOS radiation hardness assurance." Nuclear Science, IEEE Transactions on 40.6 (1993): 1678-1685.

[36] Buchner, Stephen, and Dale McMorrow. "Single-event transients in bipolar linear integrated circuits." IEEE transactions on nuclear science 53.6 (2006): 3079.

[37] A. Hastings, The Art of Analog Layout, 2nd ed., Prentice-Hall, Ed. New York, 2005.

[38] Zhou, Quming, and Kartik Mohanram. "Transistor sizing for radiation hardening." Reliability Physics Symposium Proceedings, 2004. 42nd Annual. 2004 IEEE International. IEEE, 2004.

[39] Mossawir, Benjamin, et al. "A TID and SEE radiation-hardened, wideband, low-noise amplifier." Nuclear Science, IEEE Transactions on 53.6 (2006): 3439-3448.

[40] Olson, Brian D., et al. "Analysis of parasitic PNP bipolar transistor mitigation using well contacts in 130 nm and 90 nm CMOS technology." IEEE Transactions on Nuclear Science 4.54 (2007): 894-897.

[41] Calin, Th, Michael Nicolaidis, and Raoul Velazco. "Upset hardened memory design for submicron CMOS technology." IEEE-Transactions-on-Nuclear-Science. (1996): 2874-8.

[42] J. S. Kauppila, L. W. Massengill, W. T. Holman, A. V. Kauppila, S. Sanathanamurthy, "Single event Simulation methodology for analog/mixed signal design hardening," Nuclear Science, IEEE Transactions on, vol. 51, no. 6, pp. 3603-3608, Dec. 2004.

[43] B. D. Olson, W. T. Holman, L. W. Massengill, B. L. Bhuva, "Evaluation of Radiation-Hardened Design Techniques Using Frequency Domain Analysis," Nuclear Science, IEEE Transactions on , vol. 55, no. 6, pp. 2957-2961, Dec. 2008.

[44] Duzellier, S., et al. "SEE in-flight data for two static 32KB memories on high earth orbit." Radiation Effects Data Workshop, 2002 IEEE. IEEE, 2002.

[45] Sherman, Barbara. "NASA's LWS/SET technology experiment carrier." Aerospace Conference, 2003. Proceedings. 2003 IEEE. Vol. 1. IEEE, 2003.

[46] Nugent, Ryan, et al. "The cubesat: The picosatellite standard for research and education." Aerospace Engineering 805 (2008): 756-5087.

[47] Hubert, Guillaume, Raoul Velazco, and Paul Peronnard. "A generic platform for remote accelerated tests and high altitude SEU experiments on advanced ICs: Correlation with MUSCA SEP3 calculations." International On-Line Test Symposium (IOLTS'09). IEEE Computer Society, 2009.

[48] Lesea, Austin, et al. "The rosetta experiment: atmospheric soft error rate testing in differing technology FPGAs." Device and Materials Reliability, IEEE Transactions on 5.3 (2005): 317-328.

[49] McMorrow, Dale, et al. "Application of a pulsed laser for evaluation and optimization of SEU-hard designs." Radiation and Its Effects on Components and Systems, 1999. RADECS 99. 1999 Fifth European Conference on. IEEE, 1999.

[50] Duzellier, Sophie, et al. "Application of laser testing in study of SEE mechanisms in 16-Mbit DRAMs." Nuclear Science, IEEE Transactions on 47.6 (2000): 2392-2399.

[51] Dodd, Paul E., and Lloyd W. Massengill. "Basic mechanisms and modeling of single-event upset in digital microelectronics." Nuclear Science, IEEE Transactions on 50.3 (2003): 583-602.

[52] Silvaco Radiation Effect Module, `http://www.silvaco.com/products/vwf/atlas/2D/rem/rem_br.html`

[53] Synopsys Sentaurus, `http://www.synopsys.com/Tools/TCAD/DeviceSimulation/Pages/SentaurusDevice.aspx`

[54] HEPHY, http://www.hephy.at/en/

[55] Cenna, Francesca, et al. "Weightfield2: A fast simulator for silicon and diamond solid state detector." Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 796 (2015): 149-153.

[56] ROOT, https://root.cern.ch

[57] Pérez, Fernando, and Brian E. Granger. "IPython: a system for interactive scientific computing." Computing in Science & Engineering 9.3 (2007): 21-29.

[58] Satish Balay and Shrirang Abhyankar and Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Li,ro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, Hong Zhang, PETSc Web page, http://www.mcs.anl.gov/petsc, 2015

[59] ublas, Basic Linear Algebra Library, http://www.boost.org/doc/libs/release/libs/numeric/ublas/

[60] Michael Heroux and Roscoe Bartlett and Vicki Howle Robert Hoekstra and Jonathan Hu and Tamara Kolda and Richard Lehoucq and Kevin Long and Roger Pawlowski and Eric Phipps and Andrew Salinger and Heidi Thornquist and Ray Tuminaro and James Willenbring and Alan Williams, "An Overview of Trilinos", Sandia National Laboratories,SAND2003-2927,2003

[61] MTL4, Matrix Template Library 4, http://www.simunova.com/de/node/24

[62] Logg, Anders, and Garth N. Wells. "DOLFIN: Automated finite element computing." ACM Transactions on Mathematical Software (TOMS) 37.2 (2010): 20.

[63] Alnæs, Martin Sandve, et al. "The FEniCS Manual." (2011).

[64] Rathgeber, F. L. O. R. I. A. N. Automated finite element computations in the FEniCS framework using general purpose graphics processing units. Diss. Dissertação de Mestrado, Royal Institute of Technology (KTH), Estocolmo, 2010.

[65] Pironneau, O., Hecht, F., and Le Hyaric, A. FreeFEM++. http://www.freefem.org/.

[66] Dular, P., Geuzaine, C., et al. GetDP: A general environment for the treatment of discrete problems. http://geuz.org/getdp/.

[67] Deparis, S. et al. LifeV. http://www.lifev.org/.

[68] Beazley, D. M. Automated scientific software scripting with SWIG. Future Generation Computer Systems 19, 5, 599–609. 2003

[69] Simplified Wrapper and Interface Generator (SWIG). http://www.swig.org/.

[70] Distutils. https://docs.python.org/2/distutils/introduction.html.

[71] Alnæs, Martin S., et al. "Unified Form Language: A domain-specific language for weak formulations of partial differential equations." ACM Transactions on Mathematical Software (TOMS) 40.2 (2014): 9.

[72] Phunpeng, V., and P. M. Baiz. "Mixed finite element formulations for strain-gradient elasticity problems using the FEniCS environment." Finite Elements in Analysis and Design 96 (2015): 23-40.

[73] Magnum.fe http://micromagnetics.org/magnum.fe/index.html

[74] Abert, Claas, et al. "magnum. fe: A micromagnetic finite-element simulation code based on FEniCS." Journal of Magnetism and Magnetic Materials 345 (2013): 29-35.

[75] Otto, A. J., et al. "Using the FEniCS package for FEM solutions in electromagnetics." Antennas and Propagation Magazine, IEEE 54.4 (2012): 206-223.

[76] Alioli, Mattia, Marco Morandini, and Pierangelo Masarati. "Coupled multibody-fluid dynamics simulation of flapping wings." ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, 2013.

[77] Vynnytska, L., and S. R. Clark. "FEniCS framework in geoscientific applications." MODSIM. 2011.

[78] Selim, Kristoffer, and Anders Logg. "Simulating Heart Valve Dynamics in FEniCS." MekIT 9 (2009): 26-27.

[79] Valen-Sendstad, Kristian, Marina Piccinelli, and David A. Steinman. "High-resolution computational fluid dynamics detects flow instabilities in the carotid siphon: Implications for aneurysm initiation and rupture?." Journal of biomechanics 47.12 (2014): 3210-3216.

[80] Sawicki, Bartosz, and Arkadiusz Miaskowski. "Numerical model of magnetic fluid hyperthermia." Przeglad Elektrotechniczny 89.7 (2013): 86-88.

[81] GNU Lesser General Public License `http://www.gnu.org/licenses/lgpl-3.0.en.html`

[82] P. D. Lax and A. N. Milgram. Parabolic equations. Annals of Mathematics Studies, 33:167–190, 1954.

[83] J. T. Oden and L. Demkowicz. Applied Functional Analysis. CRC press, 1996.

[84] Melenk, Jens Markus, and Barbara I. Wohlmuth. "On residual-based a posteriori error estimation in hp-FEM." Advances in Computational Mathematics 15.1-4 (2001): 311-331.

[85] Baumann, Carlos Erik, and J. Tinsley Oden. "A discontinuous hp finite element method for convection—diffusion problems." Computer Methods in Applied Mechanics and Engineering 175.3 (1999): 311-341.

[86] Bank, Randolph E., Donald J. Rose, and Wolfgang Fichtner. "Numerical methods for semiconductor device simulation." Electron Devices, IEEE Transactions on 30.9 (1983): 1031-1041.

[87] C. Geuzaine and J.-F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. International Journal for Numerical Methods in Engineering 79(11), pp. 1309-1331, 2009.

[88] "StoppingHinAlBethe", by H. Paul - Own work. Licensed under CC BY-SA 3.0 via Commons - `https://commons.wikimedia.org/wiki/File:StoppingHinAlBethe.png#/media/File:StoppingHinAlBethe.png`

[89] "Solar Flare", by NASA. Licensed under CC BY-SA 3.0 via Commons - `https://commons.wikimedia.org/wiki/File:Solar_flare_(TRACE).gif`

[90] "Solar wind", by NASA. Licensed under CC BY-SA 3.0 via Commons - https://commons.wikimedia.org/wiki/File:Structure_of_the_magnetosphere-en.svg

[91] "BraggPeak", by A. A. Miller. Licensed under CC BY-SA 3.0 via Commons - https://commons.wikimedia.org/wiki/File:BraggPeak.png#/media/File:BraggPeak.png

[92] ONELAB (Open Numerical Engineering LABoratory), http://onelab.info/wiki/ONELAB

Elenco delle figure